Test Driven Development in Assembler

a little story about growing software from nothing

Olve Maudal

During the last decade Test-Driven Development has become an established practice for developing software in the industry. All good programmers must have TDD in the toolbox so that they can use it when appropriate.

In this session I will demonstrate Test-Driven Development by example, using nothing but assembler language.

A 90 minute session
ACCU, Oxford, April 2012

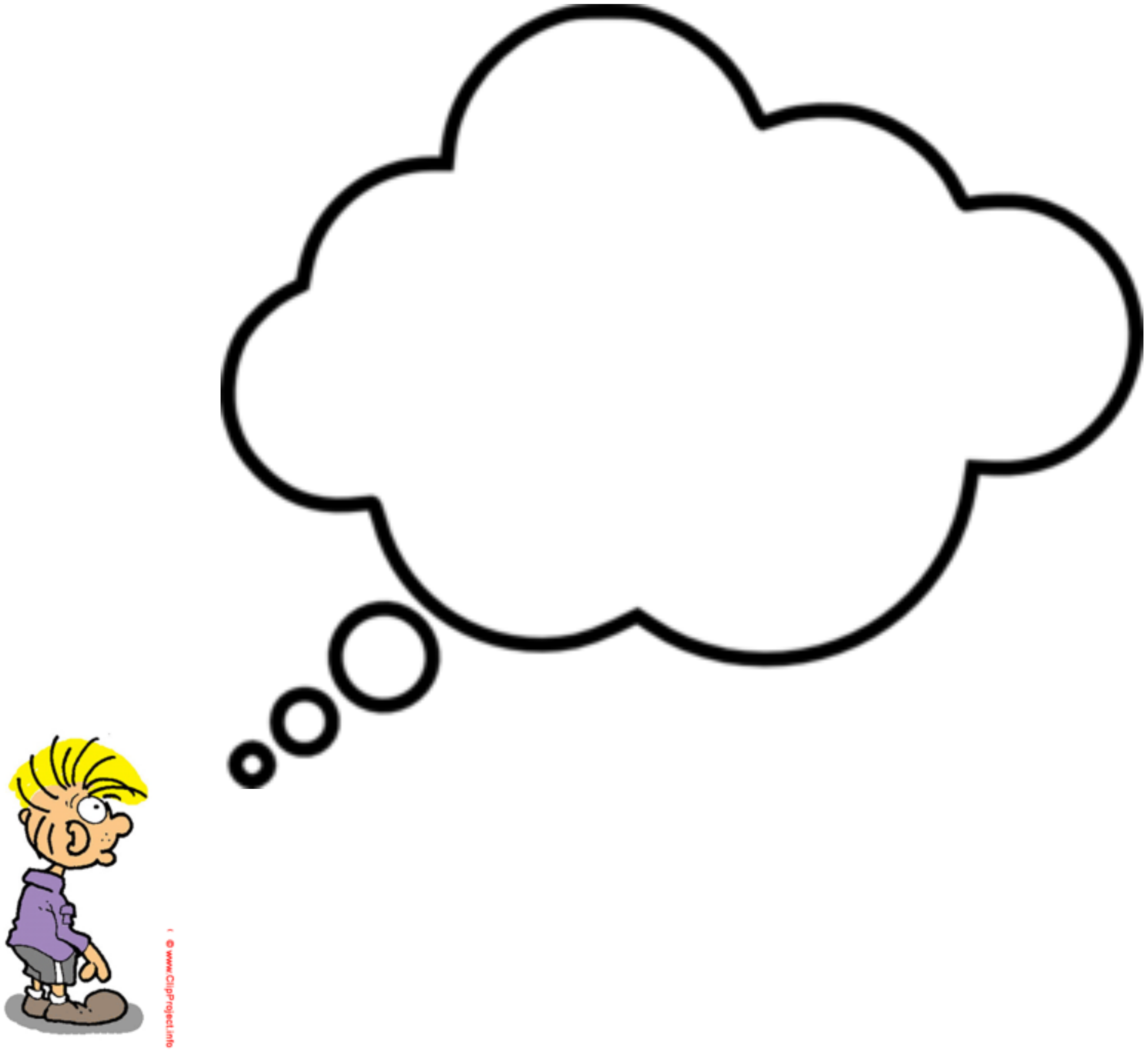
Disclaimer:

This is not meant as a tutorial to learn about assembler programming. For example, I am avoiding all/most of the idioms that experienced assembler programmers use (eg, xor to reset a variable, repeat string operations, proper looping, newer and specialized instructions etc). The reason I do this is partly because I am inexperienced with real assembly programming myself, but mostly because it does not add too much value when demonstrating TDD techniques which is the intention of this presentation.

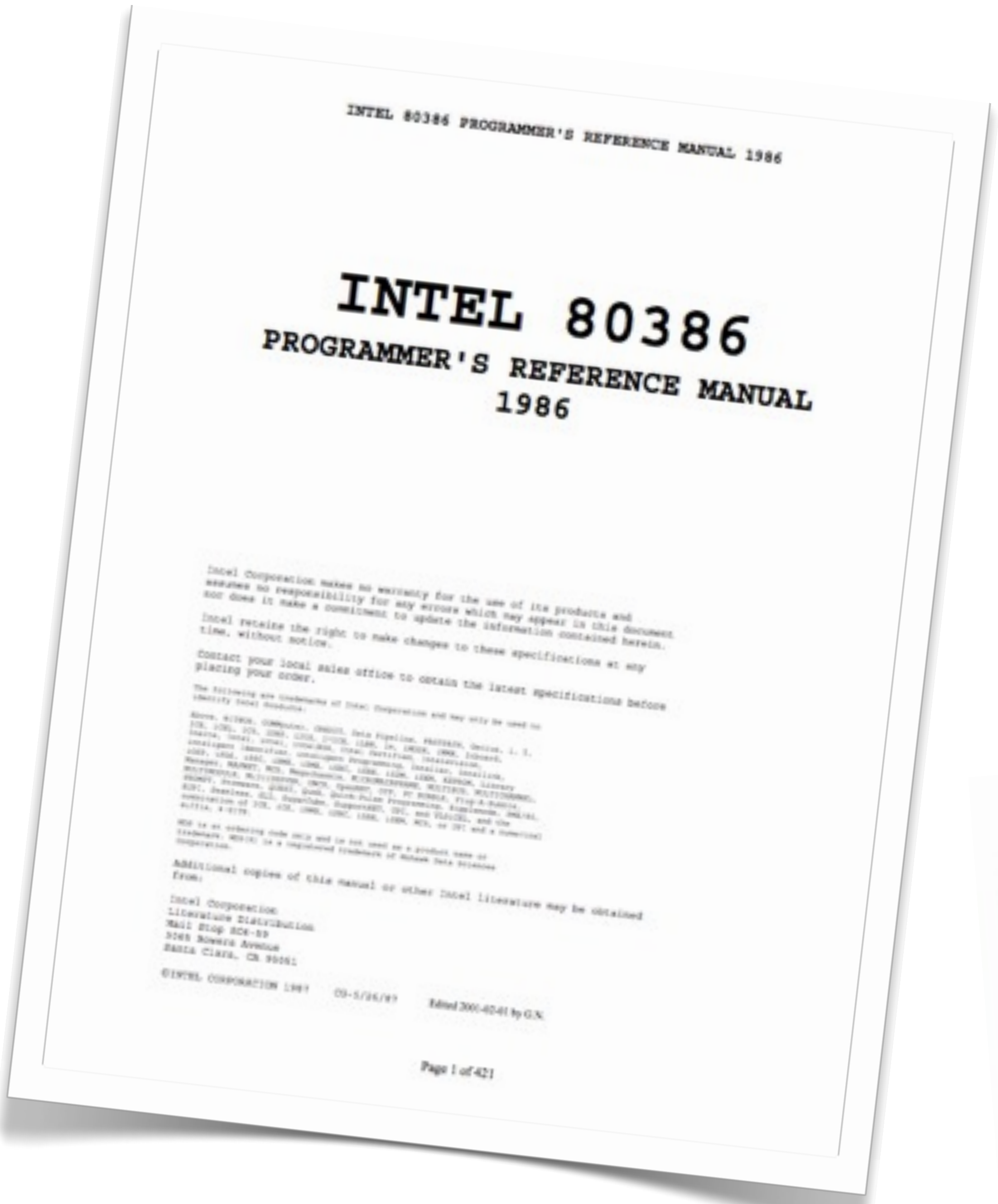
Also, I know already that there are some bugs in the code so use with care!

If you want to learn about assembler programming on Intel CPU's and BSD based systems I suggest the following sources:
http://www.drpaulcarter.com/pcasm/
http://www.int80h.org/
http://pdos.csail.mit.edu/6.858/2011/readings/i386.pdf

© www.ClipProject.info


© www.ClipProject.info

LSD *NIX i386
ed, as, ld, make
maxell
SUPER
RD
MD2-256HD


INTEL 80386 PROGRAMMER'S REFERENCE MANUAL 1986

INTEL 80386
PROGRAMMER'S REFERENCE MANUAL
1986

Intel Corporation makes no warranty for the use of its products and assumes no responsibility for any errors which may appear in this document nor does it make a commitment to update the information contained herein.

Intel retains the right to make changes to these specifications at any time, without notice.

Contact your local sales office to obtain the latest specifications before placing your order.

The following are trademarks of Intel Corporation and may only be used to identify Intel products:

Above, BITBUS, COMMputer, CREDIT, Data Pipeline, FASTPATH, Genius, i, î, ICE, iCEL, iCS, iDBP, iDIS, I²ICE, iLBX, im, iMDDX, iMMX, Inboard, Insite, Intel, intel, intelBOS, Intel Certified, Intelevision, inteligent Identifier, inteligent Programming, Intellec, Intellink, iOSP, iPDS, iPSC, iRMX, iSBC, iSBX, iSDM, iSXM, KEPROM, Library Manager, MAPNET, MCS, Megachassis, MICROMAINFRAME, MULTIBUS, MULTICHANNEL, MULTIMODULE, MultiSERVER, ONCE, OpenNET, OTP, PC BUBBLE, Plug-A-Bubble, PROMPT, Promware, QUEST, QueX, Quick-Pulse Programming, Ripplemode, RMX/80, RUPI, Seamless, SLD, SugarCube, SupportNET, UPI, and VLSiCEL, and the combination of ICE, iCS, iRMX, iSBC, iSBX, iSXM, MCS, or UPI and a numerical suffix, 4-SITE.

MDS is an ordering code only and is not used as a product name or trademark. MDS(R) is a registered trademark of Mohawk Data Sciences Corporation.

Additional copies of this manual or other Intel literature may be obtained from:

Intel Corporation
Literature Distribution
Mail Stop SC6-59
3065 Bowers Avenue
Santa Clara, CA 95051

©INTEL CORPORATION 1987   CG-5/26/87   Edited 2001-02-01 by G.N.

Page 1 of 421

Hello, world!
© www.ClipProject.info

Hello, world!

```
kernel:
        int 80h
        ret

section .data
greeting db 'Hello, world!', 0xa

section .text
global start
start:
        push dword 14           ; number of characters
        push dword greeting
        push dword 1            ; stdout
        mov eax, 4              ; sys_write
        call kernel
        add esp, 12             ; same as 3 x pop dword

        push dword 0            ; exit success value
        mov eax, 1              ; sys_exit
        call kernel
```

```
$ uname -v
Darwin Kernel Version 10.8.0: Tue Jun  7 16:33:36 PDT 2011;
root:xnu-1504.15.3~1/RELEASE_I386
$ nasm -f macho hello.asm
$ ld -o hello hello.o
$ ./hello
Hello, world!
$ echo $?
0
$
```

Hello, world!

```
kernel:
        int 80h
        ret

%define SYS_EXIT 1
%define SYS_READ 3
%define SYS_WRITE 4

%define STDIN 0
%define STDOUT 1
%define STDERR 2

%define EXIT_SUCCESS 0
%define EXIT_FAILURE 1

section .data
greeting db 'Hello, world!', 0xa
greeting_len equ $-greeting

section .text
global start
start:
        push dword greeting_len
        push dword greeting
        push dword STDOUT
        mov eax, SYS_WRITE
        call kernel
        add esp, 12

        push dword EXIT_SUCCESS
        mov eax, SYS_EXIT
        call kernel
```

Hello, world!

mylib.inc

```
kernel:
        int 0x80
        ret

%define SYS_EXIT 1
%define SYS_READ 3
%define SYS_WRITE 4

%define STDIN 0
%define STDOUT 1
%define STDERR 2

%define EXIT_SUCCESS 0
%define EXIT_FAILURE 1

%macro sys_exit 1
        push dword %1
        mov eax, SYS_EXIT
        call kernel
%endmacro

%macro sys_read 3
        push dword %3
        push dword %2
        push dword %1
        mov eax, SYS_READ
        call kernel
        add esp, 12
%endmacro

%macro sys_write 3
        push dword %3
        push dword %2
        push dword %1
        mov eax, SYS_WRITE
        call kernel
        add esp, 12
%endmacro
```

hello.asm

```
%include "mylib.inc"

section .data
greeting db 'Hello, world!', 0xa
greeting_len equ $-greeting

section .text
global start
start:
        sys_write STDOUT, greeting, greeting_len
        sys_exit EXIT_SUCCESS
```

```
%include "mylib.inc"

%define BUFFERSIZE 1024
global start
start:
section .data
        .str1 db "What is your name? "
        .len1 equ $-.str1
        .str2 db " is invincible!", 0xa
        .len2 equ $-.str2
        .buflen dd 0
section .bss
        .buf resb BUFFERSIZE
section .text
        sys_write STDOUT, .str1, .len1
        sys_read STDIN, .buf, BUFFERSIZE
        cmp eax, 0
        jl .exit_with_failure
        je .exit_with_success
        mov [.buflen], eax
        sub [.buflen], dword 1
        mov ecx, 10
.again:
        sys_write STDOUT, .buf, [.buflen]
        sys_write STDOUT, .str2, .len2
        dec ecx
        jnz .again
.exit_with_success:
        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

```
%include "mylib.inc"

%define BUFFERSIZE 1024
global start
start:
section .data
        .str1 db "What is your name? "
        .len1 equ $-.str1
        .str2 db " is invincible!", 0xa
        .len2 equ $-.str2
        .buflen dd 0
section .bss
        .buf resb BUFFERSIZE
section .text
        sys_write STDOUT, .str1, .len1
        sys_read STDIN, .buf, BUFFERSIZE
        cmp eax, 0
        jl .exit_with_failure
        je .exit_with_success
        mov [.buflen], eax
        sub [.buflen], dword 1
        mov ecx, 10
.again:
        sys_write STDOUT, .buf, [.buflen]
        sys_write STDOUT, .str2, .len2
        dec ecx
        jnz .again
.exit_with_success:
        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

```
$ nasm -f macho askme.asm
```

```
%include "mylib.inc"

%define BUFFERSIZE 1024
global start
start:
section .data
        .str1 db "What is your name? "
        .len1 equ $-.str1
        .str2 db " is invincible!", 0xa
        .len2 equ $-.str2
        .buflen dd 0
section .bss
        .buf resb BUFFERSIZE
section .text
        sys_write STDOUT, .str1, .len1
        sys_read STDIN, .buf, BUFFERSIZE
        cmp eax, 0
        jl .exit_with_failure
        je .exit_with_success
        mov [.buflen], eax
        sub [.buflen], dword 1
        mov ecx, 10
.again:
        sys_write STDOUT, .buf, [.buflen]
        sys_write STDOUT, .str2, .len2
        dec ecx
        jnz .again
.exit_with_success:
        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

```
$ nasm -f macho askme.asm
$ ld askme.o
```

```
%include "mylib.inc"

%define BUFFERSIZE 1024
global start
start:
section .data
        .str1 db "What is your name? "
        .len1 equ $-.str1
        .str2 db " is invincible!", 0xa
        .len2 equ $-.str2
        .buflen dd 0
section .bss
        .buf resb BUFFERSIZE
section .text
        sys_write STDOUT, .str1, .len1
        sys_read STDIN, .buf, BUFFERSIZE
        cmp eax, 0
        jl .exit_with_failure
        je .exit_with_success
        mov [.buflen], eax
        sub [.buflen], dword 1
        mov ecx, 10
.again:
        sys_write STDOUT, .buf, [.buflen]
        sys_write STDOUT, .str2, .len2
        dec ecx
        jnz .again
.exit_with_success:
        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

```
$ nasm -f macho askme.asm
$ ld askme.o
$ ./a.out
```

```
%include "mylib.inc"

%define BUFFERSIZE 1024
global start
start:
section .data
        .str1 db "What is your name? "
        .len1 equ $-.str1
        .str2 db " is invincible!", 0xa
        .len2 equ $-.str2
        .buflen dd 0
section .bss
        .buf resb BUFFERSIZE
section .text
        sys_write STDOUT, .str1, .len1
        sys_read STDIN, .buf, BUFFERSIZE
        cmp eax, 0
        jl .exit_with_failure
        je .exit_with_success
        mov [.buflen], eax
        sub [.buflen], dword 1
        mov ecx, 10
.again:
        sys_write STDOUT, .buf, [.buflen]
        sys_write STDOUT, .str2, .len2
        dec ecx
        jnz .again
.exit_with_success:
        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

```
$ nasm -f macho askme.asm
$ ld askme.o
$ ./a.out
What is your name?
```

```
%include "mylib.inc"

%define BUFFERSIZE 1024
global start
start:
section .data
        .str1 db "What is your name? "
        .len1 equ $-.str1
        .str2 db " is invincible!", 0xa
        .len2 equ $-.str2
        .buflen dd 0
section .bss
        .buf resb BUFFERSIZE
section .text
        sys_write STDOUT, .str1, .len1
        sys_read STDIN, .buf, BUFFERSIZE
        cmp eax, 0
        jl .exit_with_failure
        je .exit_with_success
        mov [.buflen], eax
        sub [.buflen], dword 1
        mov ecx, 10
.again:
        sys_write STDOUT, .buf, [.buflen]
        sys_write STDOUT, .str2, .len2
        dec ecx
        jnz .again
.exit_with_success:
        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

```
$ nasm -f macho askme.asm
$ ld askme.o
$ ./a.out
What is your name? Larry
```

```
%include "mylib.inc"

%define BUFFERSIZE 1024
global start
start:
section .data
        .str1 db "What is your name? "
        .len1 equ $-.str1
        .str2 db " is invincible!", 0xa
        .len2 equ $-.str2
        .buflen dd 0
section .bss
        .buf resb BUFFERSIZE
section .text
        sys_write STDOUT, .str1, .len1
        sys_read STDIN, .buf, BUFFERSIZE
        cmp eax, 0
        jl .exit_with_failure
        je .exit_with_success
        mov [.buflen], eax
        sub [.buflen], dword 1
        mov ecx, 10
.again:
        sys_write STDOUT, .buf, [.buflen]
        sys_write STDOUT, .str2, .len2
        dec ecx
        jnz .again
.exit_with_success:
        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

```
$ nasm -f macho askme.asm
$ ld askme.o
$ ./a.out
What is your name? Larry
Larry is invincible!
Larry is invincible!
Larry is invincible!
Larry is invincible!
Larry is invincible!
Larry is invincible!
Larry is invincible!
Larry is invincible!
Larry is invincible!
Larry is invincible!
$
```

Monday

Monday

Hey, programmer. I got an idea. I need some stuff...

Monday

Hey, programmer. I got an idea. I need some stuff...

ok?

Monday

Hey, programmer. I got an idea. I need some stuff...

ok?

a program that can calculate the scores in a dice game.

Monday

Hey, programmer. I got an idea. I need some stuff...

ok?

a program that can calculate the scores in a dice game.

which dice game?

Monday

Hey, programmer. I got an idea. I need some stuff...

ok?

a program that can calculate the scores in a dice game.

which dice game?

I need it by friday

Monday

Hey, programmer. I got an idea. I need some stuff...

ok?

a program that can calculate the scores in a dice game.

which dice game?

is it yahtzee?

I need it by friday

Monday

Hey, programmer. I got an idea. I need some stuff...

ok?

a program that can calculate the scores in a dice game.

which dice game?

I need it by friday

is it yahtzee?

yeah, something like that, whatever, original american version? You only need to care about the lower section...you figure it out?

Monday

Hey, programmer. I got an idea. I need some stuff...

ok?

a program that can calculate the scores in a dice game.

which dice game?

I need it by friday

is it yahtzee?

yeah, something like that, whatever, original american version? You only need to care about the lower section...you figure it out?

I guess so... but what kind of input do I get? And which format of output do you expect? And what is it going to be used for?

Monday

Hey, programmer. I got an idea. I need some stuff...

ok?

a program that can calculate the scores in a dice game.

which dice game?

I need it by friday

is it yahtzee?

yeah, something like that, whatever, original american version? You only need to care about the lower section...you figure it out?

I guess so... but what kind of input do I get? And which format of output do you expect? And what is it going to be used for?

Input, output, bits and bytes? **GEEK**! Why ask me? you are the programmer

Monday

Hey, programmer. I got an idea. I need some stuff...

ok?

a program that can calculate the scores in a dice game.

which dice game?

I need it by friday

is it yahtzee?

yeah, something like that, whatever, original american version? You only need to care about the lower section...you figure it out?

I guess so... but what kind of input do I get? And which format of output do you expect? And what is it going to be used for?

Input, output, bits and bytes? **GEEK**! Why ask me? you are the programmer

eat flaming death!

Monday

Hey, programmer. I got an idea. I need some stuff...

ok?

a program that can calculate the scores in a dice game.

which dice game?

I need it by friday

is it yahtzee?

yeah, something like that, whatever, original american version? You only need to care about the lower section...you figure it out?

I guess so... but what kind of input do I get? And which format of output do you expect? And what is it going to be used for?

Input, output, bits and bytes? **GEEK**! Why ask me? you are the programmer

eat flaming death!

I need it by friday. Right?

Monday

Hey, programmer. I got an idea. I need some stuff...

ok?

a program that can calculate the scores in a dice game.

which dice game?

I need it by friday

is it yahtzee?

yeah, something like that, whatever, original american version? You only need to care about the lower section...you figure it out?

I guess so... but what kind of input do I get? And which format of output do you expect? And what is it going to be used for?

Input, output, bits and bytes? **GEEK**! Why ask me? you are the programmer

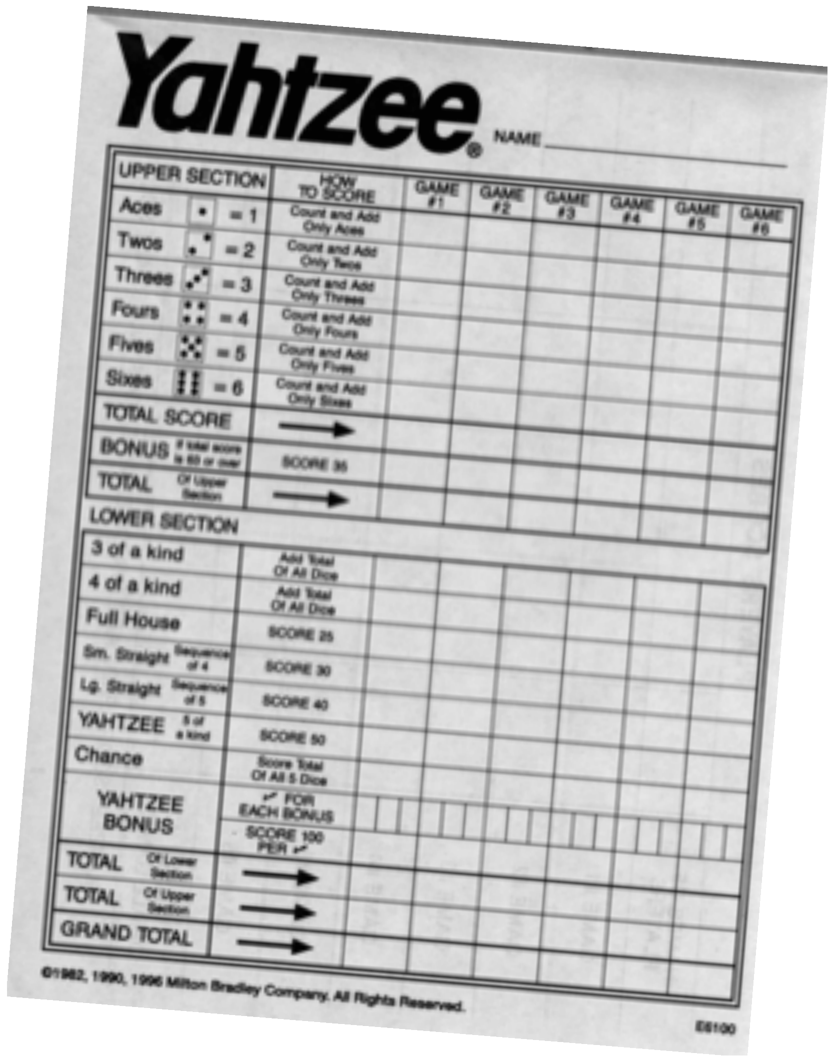
eat flaming death!

I need it by friday. Right?

sure

vague initial requirement

Write a library that can score the lower section of a game of yahtzee.

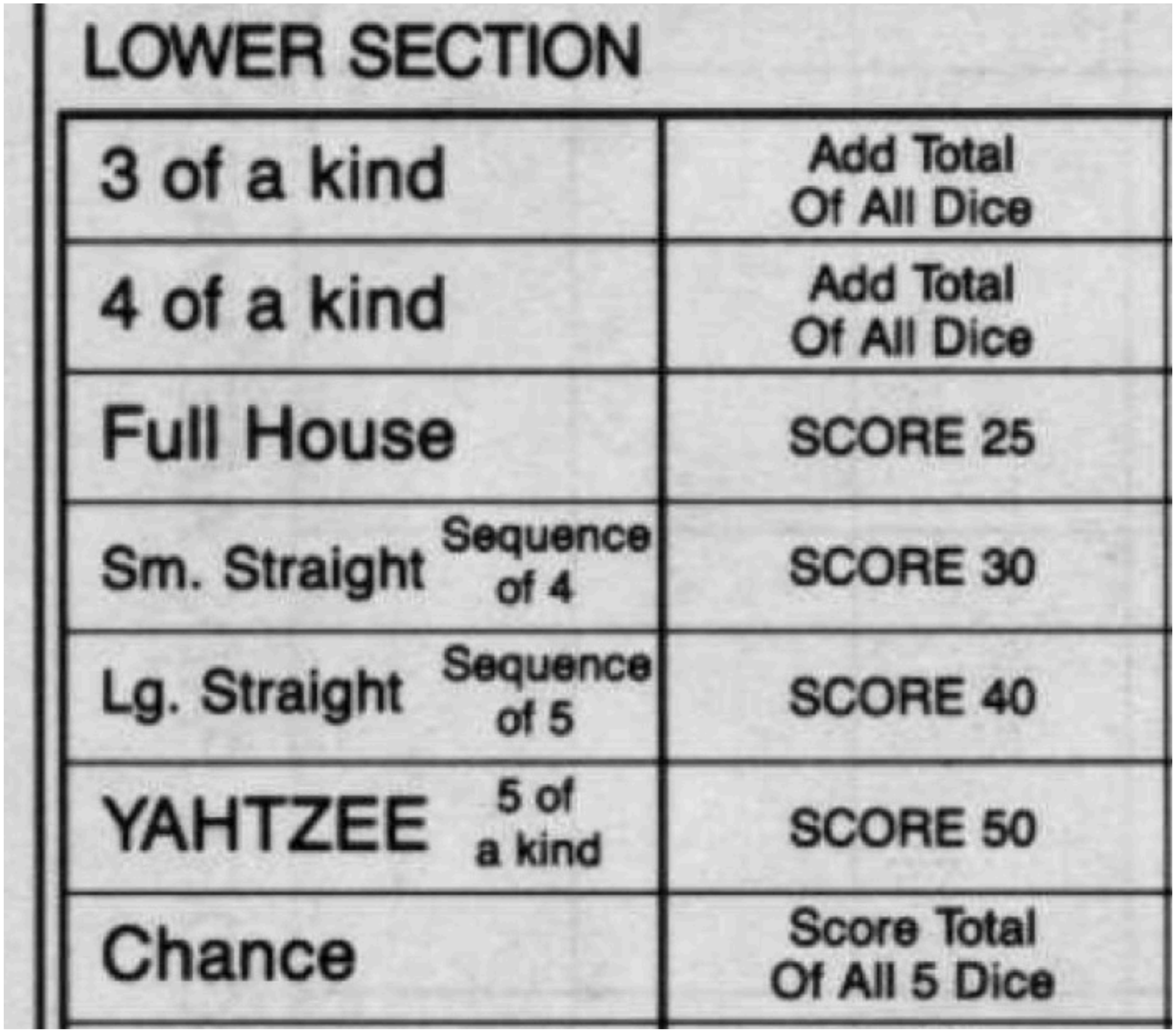


Yahtzee® NAME ____________

UPPER SECTION	HOW TO SCORE	GAME #1	GAME #2	GAME #3	GAME #4	GAME #5	GAME #6
Aces ⚀ = 1	Count and Add Only Aces						
Twos ⚁ = 2	Count and Add Only Twos						
Threes ⚂ = 3	Count and Add Only Threes						
Fours ⚃ = 4	Count and Add Only Fours						
Fives ⚄ = 5	Count and Add Only Fives						
Sixes ⚅ = 6	Count and Add Only Sixes						
TOTAL SCORE	→						
BONUS If total score is 63 or over	SCORE 35						
TOTAL Of Upper Section	→						
LOWER SECTION							
3 of a kind	Add Total Of All Dice						
4 of a kind	Add Total Of All Dice						
Full House	SCORE 25						
Sm. Straight Sequence of 4	SCORE 30						
Lg. Straight Sequence of 5	SCORE 40						
YAHTZEE 5 of a kind	SCORE 50						
Chance	Score Total Of All 5 Dice						
YAHTZEE BONUS	✓ FOR EACH BONUS						
	SCORE 100 PER ✓						
TOTAL Of Lower Section	→						
TOTAL Of Upper Section	→						
GRAND TOTAL	→						

©1982, 1990, 1996 Milton Bradley Company. All Rights Reserved.

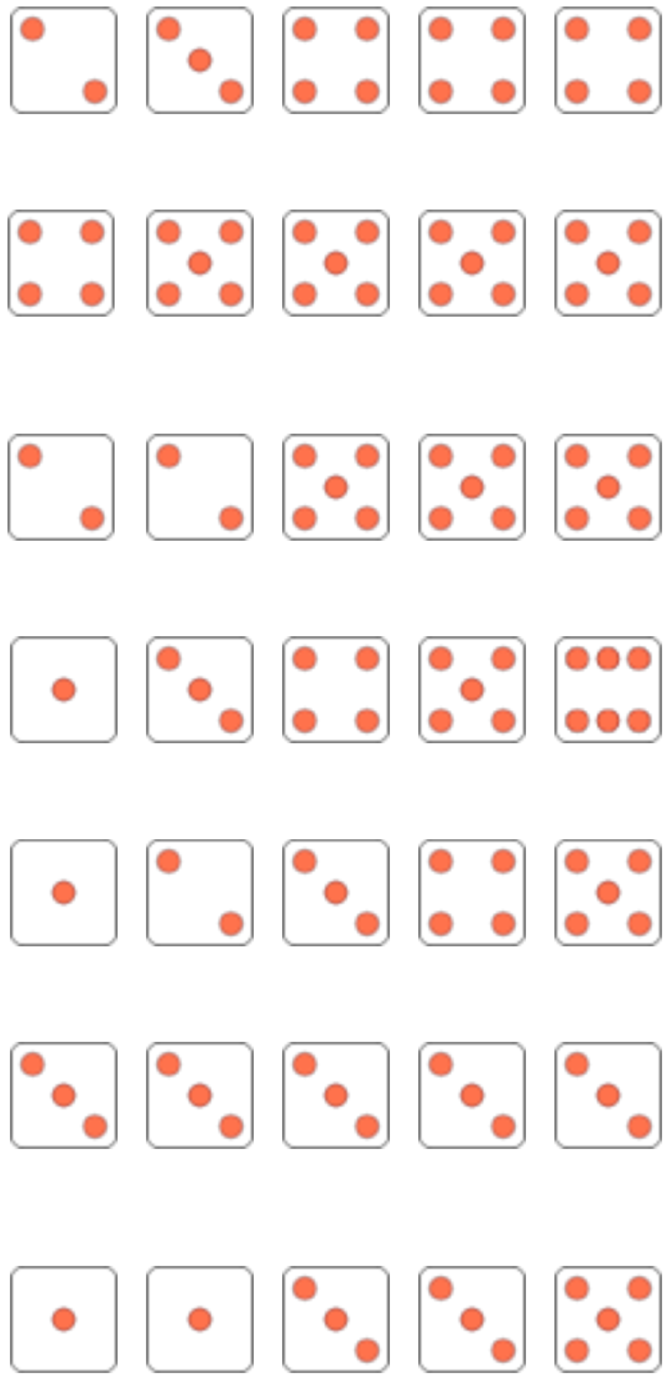
E6100



LOWER SECTION

3 of a kind	Add Total Of All Dice
4 of a kind	Add Total Of All Dice
Full House	SCORE 25
Sm. Straight Sequence of 4	SCORE 30
Lg. Straight Sequence of 5	SCORE 40
YAHTZEE 5 of a kind	SCORE 50
Chance	Score Total Of All 5 Dice

Examples



(see also http://en.wikipedia.org/wiki/Yahtzee)

$$\begin{tikzcd}
\texttt{yahtzee.asm} \arrow[dr, "\text{nasm}"] & & & \texttt{yahtzee\_tests.asm} \arrow[d, "\text{nasm}"] \\
& \texttt{yahtzee.o} \arrow[dr, "\text{ar}"] & & \texttt{yahtzee\_tests.o} \arrow[dd] \\
& & \texttt{yahtzee.a} \arrow[dr, "\text{ld}"] & \\
& & & \texttt{yahtzee\_tests}
\end{tikzcd}$$

Makefile

```
all: yahtzee.a

check: yahtzee_tests
	./yahtzee_tests

yahtzee.a: yahtzee.o
	ar -rcs $@ $^

yahtzee.o: yahtzee.asm mylib.inc
	nasm -f macho $<

yahtzee_tests.o: yahtzee_tests.asm mylib.inc
	nasm -f macho $<

yahtzee_tests: yahtzee_tests.o yahtzee.a
	ld -o $@ $^

clean:
	rm -f a.out *.o *.a yahtzee_tests
```

yahtzee_tests.asm

yahtzee_tests.asm

```
%include "mylib.inc"

global start
start:
        sys_exit EXIT_SUCCESS
```

yahtzee_tests.asm

```
%include "mylib.inc"

global start
start:
        sys_exit EXIT_SUCCESS
```

```
make check
```

yahtzee_tests.asm

```
%include "mylib.inc"

global start
start:
        sys_exit EXIT_SUCCESS
```

```
make check
nasm -f macho yahtzee_tests.asm
```

yahtzee_tests.asm

```
%include "mylib.inc"

global start
start:
        sys_exit EXIT_SUCCESS
```

```
make check
nasm -f macho yahtzee_tests.asm
nasm -f macho yahtzee.asm
```

yahtzee_tests.asm

```
%include "mylib.inc"

global start
start:
        sys_exit EXIT_SUCCESS
```

```
make check
nasm -f macho yahtzee_tests.asm
nasm -f macho yahtzee.asm
ar -rcs yahtzee.a yahtzee.o
```

yahtzee_tests.asm

```
%include "mylib.inc"

global start
start:
        sys_exit EXIT_SUCCESS
```

```
make check
nasm -f macho yahtzee_tests.asm
nasm -f macho yahtzee.asm
ar -rcs yahtzee.a yahtzee.o
ld -o yahtzee_tests yahtzee_tests.o yahtzee.a
```

yahtzee_tests.asm

```
%include "mylib.inc"

global start
start:
        sys_exit EXIT_SUCCESS
```

```
make check
nasm -f macho yahtzee_tests.asm
nasm -f macho yahtzee.asm
ar -rcs yahtzee.a yahtzee.o
ld -o yahtzee_tests yahtzee_tests.o yahtzee.a
./yahtzee_tests
```

yahtzee_tests.asm

```
%include "mylib.inc"

global start
start:
        sys_exit EXIT_SUCCESS
```

```
make check
nasm -f macho yahtzee_tests.asm
nasm -f macho yahtzee.asm
ar -rcs yahtzee.a yahtzee.o
ld -o yahtzee_tests yahtzee_tests.o yahtzee.a
./yahtzee_tests
bash-3.2$ echo $?
```

yahtzee_tests.asm

```
%include "mylib.inc"

global start
start:
        sys_exit EXIT_SUCCESS
```

```
make check
nasm -f macho yahtzee_tests.asm
nasm -f macho yahtzee.asm
ar -rcs yahtzee.a yahtzee.o
ld -o yahtzee_tests yahtzee_tests.o yahtzee.a
./yahtzee_tests
bash-3.2$ echo $?
0
```

yahtzee_tests.asm

```
%include "mylib.inc"

global start
start:
        sys_exit EXIT_SUCCESS
```

```
make check
nasm -f macho yahtzee_tests.asm
nasm -f macho yahtzee.asm
ar -rcs yahtzee.a yahtzee.o
ld -o yahtzee_tests yahtzee_tests.o yahtzee.a
./yahtzee_tests
bash-3.2$ echo $?
0
bash-3.2$
```

yahtzee_tests.asm

```
%include "mylib.inc"

global start
start:
        mov eax, 6
        imul eax, 9
        cmp eax, 42
        jne .exit_with_failure
        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

yahtzee_tests.asm

```
%include "mylib.inc"

global start
start:
        mov eax, 6
        imul eax, 9
        cmp eax, 42
        jne .exit_with_failure
        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

```
$ make check
```

yahtzee_tests.asm

```
%include "mylib.inc"

global start
start:
        mov eax, 6
        imul eax, 9
        cmp eax, 42
        jne .exit_with_failure
        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

```
$ make check
nasm -f macho yahtzee_tests.asm
```

yahtzee_tests.asm

```
%include "mylib.inc"

global start
start:
        mov eax, 6
        imul eax, 9
        cmp eax, 42
        jne .exit_with_failure
        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

```
$ make check
nasm -f macho yahtzee_tests.asm
nasm -f macho yahtzee.asm
```

yahtzee_tests.asm

```
%include "mylib.inc"

global start
start:
        mov eax, 6
        imul eax, 9
        cmp eax, 42
        jne .exit_with_failure
        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

```
$ make check
nasm -f macho yahtzee_tests.asm
nasm -f macho yahtzee.asm
ar -rcs yahtzee.a yahtzee.o
```

yahtzee_tests.asm

```
%include "mylib.inc"

global start
start:
        mov eax, 6
        imul eax, 9
        cmp eax, 42
        jne .exit_with_failure
        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

```
$ make check
nasm -f macho yahtzee_tests.asm
nasm -f macho yahtzee.asm
ar -rcs yahtzee.a yahtzee.o
ld -o yahtzee_tests yahtzee_tests.o yahtzee.a
```

yahtzee_tests.asm

```
%include "mylib.inc"

global start
start:
        mov eax, 6
        imul eax, 9
        cmp eax, 42
        jne .exit_with_failure
        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

```
$ make check
nasm -f macho yahtzee_tests.asm
nasm -f macho yahtzee.asm
ar -rcs yahtzee.a yahtzee.o
ld -o yahtzee_tests yahtzee_tests.o yahtzee.a
./yahtzee_tests
```

yahtzee_tests.asm

```
%include "mylib.inc"

global start
start:
        mov eax, 6
        imul eax, 9
        cmp eax, 42
        jne .exit_with_failure
        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

```
$ make check
nasm -f macho yahtzee_tests.asm
nasm -f macho yahtzee.asm
ar -rcs yahtzee.a yahtzee.o
ld -o yahtzee_tests yahtzee_tests.o yahtzee.a
./yahtzee_tests
make: *** [check] Error 1
```

yahtzee_tests.asm

```
%include "mylib.inc"

global start
start:
        mov eax, 6
        imul eax, 9
        cmp eax, 42
        jne .exit_with_failure
        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

```
$ make check
nasm -f macho yahtzee_tests.asm
nasm -f macho yahtzee.asm
ar -rcs yahtzee.a yahtzee.o
ld -o yahtzee_tests yahtzee_tests.o yahtzee.a
./yahtzee_tests
make: *** [check] Error 1
$
```

yahtzee_tests.asm

```
%include "mylib.inc"

global start
start:
        mov eax, 6
        imul eax, 9
        cmp eax, 42
        jne .exit_with_failure
        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

```
$ make check
nasm -f macho yahtzee_te
nasm -f macho yahtzee.as
ar -rcs yahtzee.a yahtz   .o
ld -o yahtzee_tests y  tzee_tests.o yahtzee.a
./yahtzee_tests
make: *** [check] Error 1
$
```

Our first unit test failed. This is good! Exactly what we want. The rythm of TDD is : fail, fix, pass... fail, fix, pass

yahtzee_tests.asm

```
%include "mylib.inc"

global start
start:
        mov eax, 6
        imul eax, 9
        cmp eax, 42
        jne .exit_with_failure
        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

```
$ make check
nasm -f macho yahtzee_te
           cho yahtzee.as
           htzee.a yahtz    .o
ld -o yahtzee_tests yahtzee_tests.o yahtzee.a
./yahtzee_tests
make: *** [check] Error 1
$
```

Fail - Fix - Pass

Our first unit test failed. This is good! Exactly what we want. The rythm of TDD is : fail, fix, pass... fail, fix, pass

yahtzee_tests.asm

```
%include "mylib.inc"

global start
start:
        mov eax, 6
        imul eax, 9
        cmp eax, 42
        jne .exit_with_failure
        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

```
$ make check
nasm -f macho yahtzee_te
               cho yahtzee.as
               ahtzee.a yahtz    .o
ld -o ya tzee_tests ya tzee_tests.o yahtzee.a
./yahtze _tests
make: *** [check] Error 1
$
```

Fail - Fix - Pass

Our first unit test failed. This is good! Exactly what we want. The rythm of TDD is : fail, fix, pass... fail, fix, pass

yahtzee_tests.asm

```
%include "mylib.inc"

global start
start:
        mov eax, 6
        imul eax, 7
        cmp eax, 42
        jne .exit_with_failure
        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

Fail - **Fix** - Pass

yahtzee_tests.asm

```
%include "mylib.inc"

global start
start:
        mov eax, 6
        imul eax, 7
        cmp eax, 42
        jne .exit_with_failure
        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

Fail - **Fix** - Pass

```
$ make check
```

yahtzee_tests.asm

```
%include "mylib.inc"

global start
start:
        mov eax, 6
        imul eax, 7
        cmp eax, 42
        jne .exit_with_failure
        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

Fail - **Fix** - Pass

```
$ make check
nasm -f macho yahtzee_tests.asm
```

yahtzee_tests.asm

```
%include "mylib.inc"

global start
start:
        mov eax, 6
        imul eax, 7
        cmp eax, 42
        jne .exit_with_failure
        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

Fail - **Fix** - Pass

```
$ make check
nasm -f macho yahtzee_tests.asm
ld -o yahtzee_tests yahtzee_tests.o yahtzee.a
```

yahtzee_tests.asm

```
%include "mylib.inc"

global start
start:
        mov eax, 6
        imul eax, 7
        cmp eax, 42
        jne .exit_with_failure
        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

Fail - **Fix** - Pass

```
$ make check
nasm -f macho yahtzee_tests.asm
ld -o yahtzee_tests yahtzee_tests.o yahtzee.a
./yahtzee_tests
```

yahtzee_tests.asm

```
%include "mylib.inc"

global start
start:
        mov eax, 6
        imul eax, 7
        cmp eax, 42
        jne .exit_with_failure
        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

Fail - **Fix** - Pass

```
$ make check
nasm -f macho yahtzee_tests.asm
ld -o yahtzee_tests yahtzee_tests.o yahtzee.a
./yahtzee_tests
$
```

yahtzee_tests.asm

```
%include "mylib.inc"

global start
start:
        mov eax, 6
        imul eax, 7
        cmp eax, 42
        jne .exit_with_failure
        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

Fail - **Fix** - Pass

Fail - Fix - **Pass**

```
$ make check
nasm -f macho yahtzee_tests.asm
ld -o yahtzee_tests yahtzee_tests.o yahtzee.a
./yahtzee_tests
$
```

yahtzee_tests.asm

```
%include "mylib.inc"

global start
start:
        mov eax, 6
        imul eax, 7
        cmp eax, 42
        jne .exit_with_failure
        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

Fail - **Fix** - Pass

Fail - Fix - **Pass**

```
$ make check
nasm -f macho yahtzee_tests.asm
ld -o yahtzee_tests yahtzee_tests.o yahtzee.a
./yahtzee_tests
$
```

All tests are OK!

Let's write a proper unit test

yahtzee_tests.asm

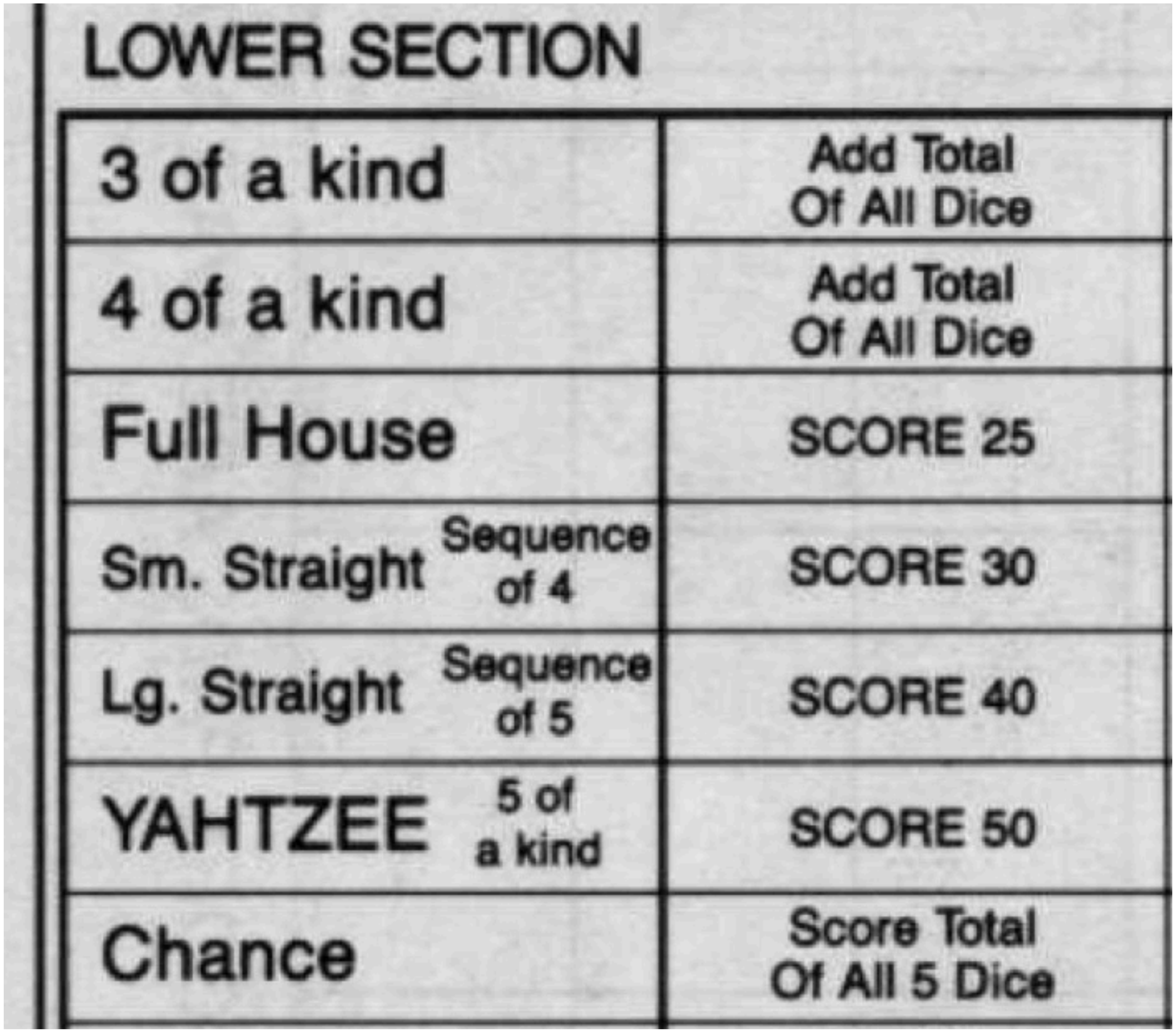
```
%include "mylib.inc"

global start
start:
        mov eax, 6
        imul eax, 7
        cmp eax, 42
        jne .exit_with_failure
        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

Fail - **Fix** - Pass

Fail - Fix - **Pass**

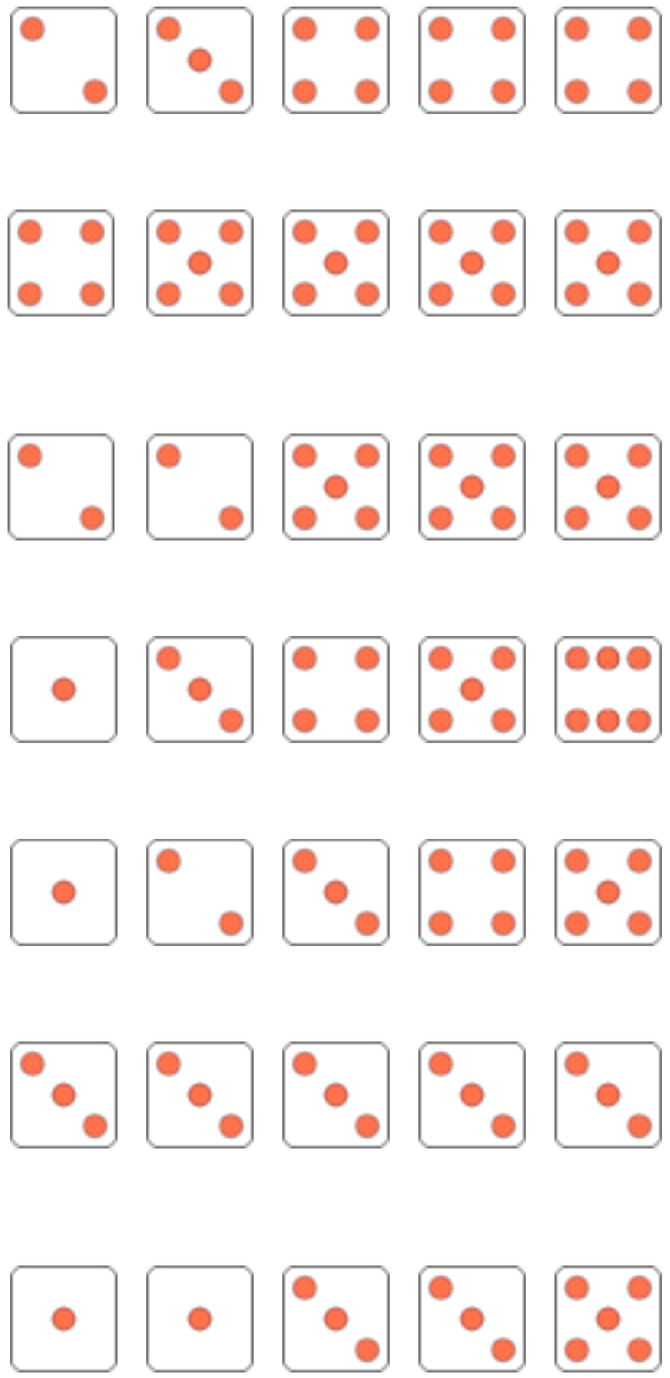
```
$ make check
nasm -f macho yahtzee_tests.asm
ld -o yahtzee_tests yahtzee_tests.o yahtzee.a
./yahtzee_tests
$
```

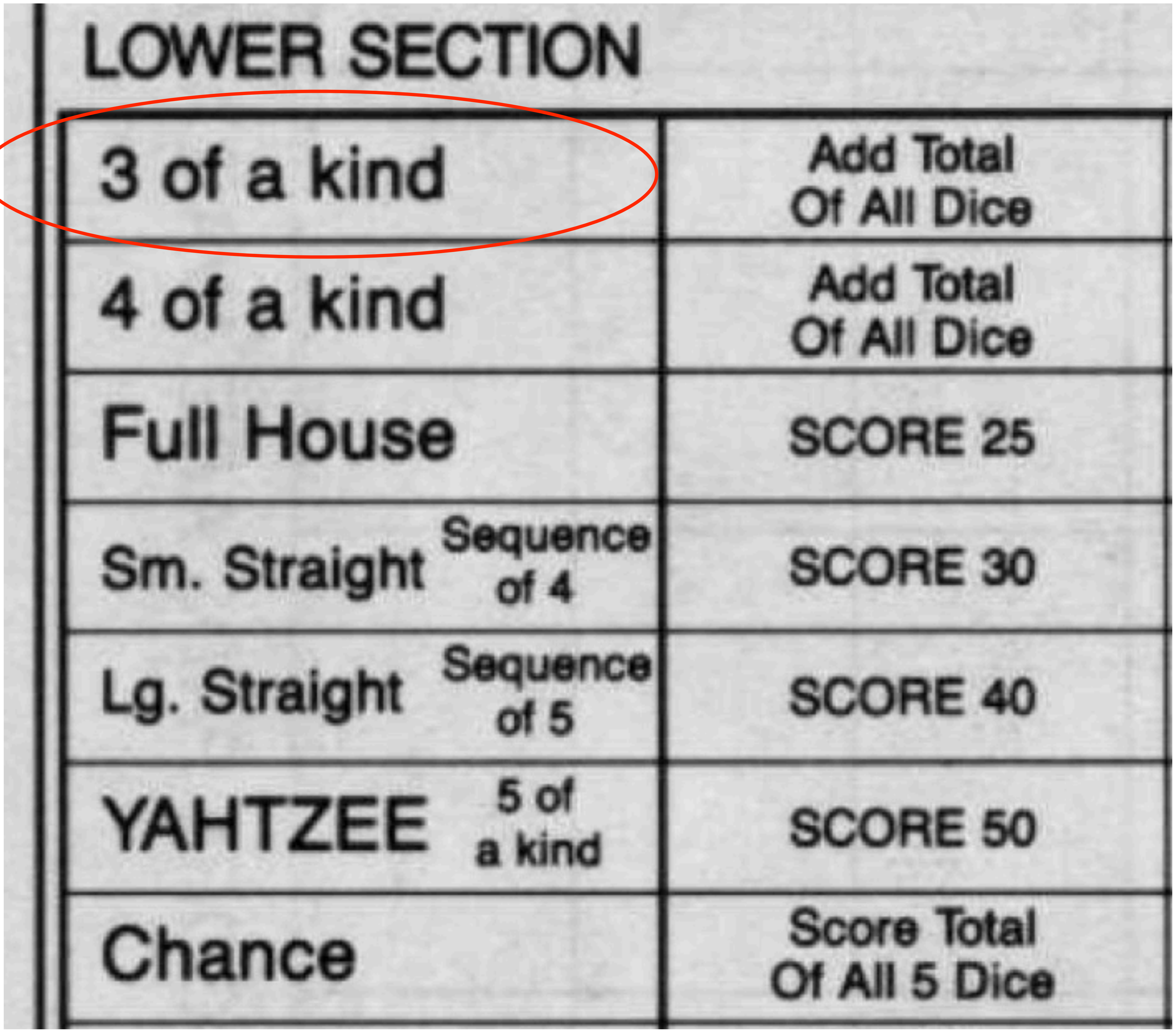
All tests are OK!



LOWER SECTION

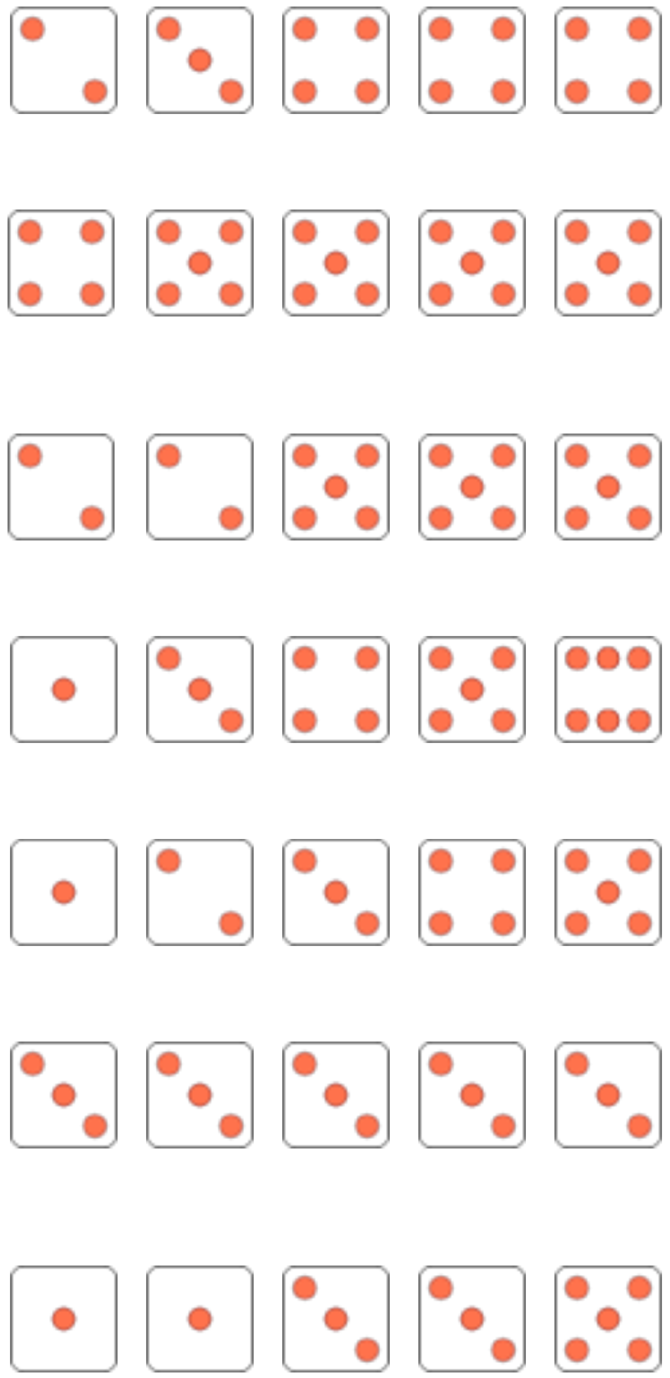
3 of a kind		Add Total Of All Dice
4 of a kind		Add Total Of All Dice
Full House		SCORE 25
Sm. Straight	Sequence of 4	SCORE 30
Lg. Straight	Sequence of 5	SCORE 40
YAHTZEE	5 of a kind	SCORE 50
Chance		Score Total Of All 5 Dice

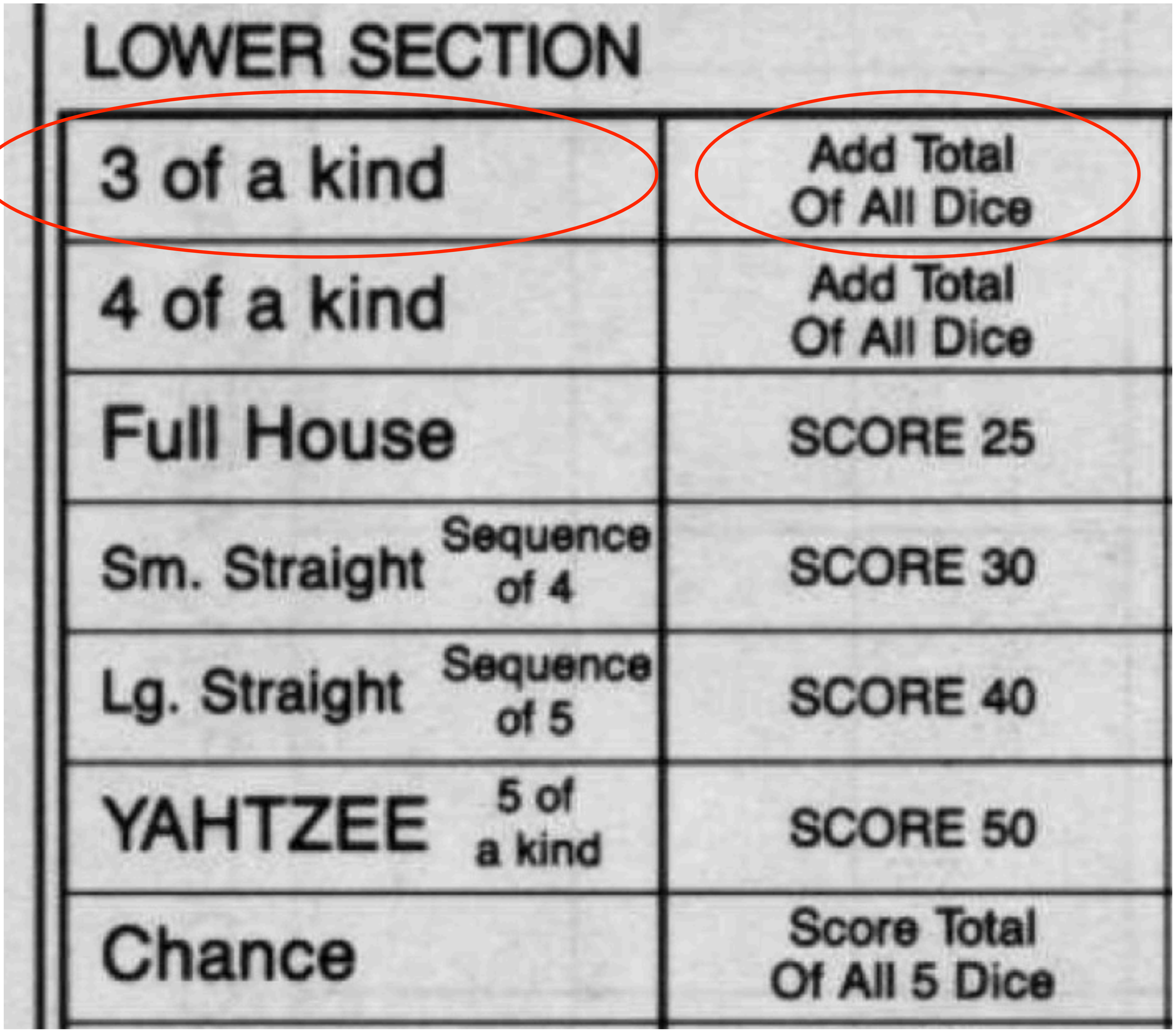




LOWER SECTION

3 of a kind		Add Total Of All Dice
4 of a kind		Add Total Of All Dice
Full House		SCORE 25
Sm. Straight	Sequence of 4	SCORE 30
Lg. Straight	Sequence of 5	SCORE 40
YAHTZEE	5 of a kind	SCORE 50
Chance		Score Total Of All 5 Dice





LOWER SECTION

3 of a kind	Add Total Of All Dice	⚁ ⚂ ⚃ ⚃ ⚃
4 of a kind	Add Total Of All Dice	⚃ ⚄ ⚄ ⚄ ⚄
Full House	SCORE 25	⚁ ⚁ ⚄ ⚄ ⚄
Sm. Straight Sequence of 4	SCORE 30	⚀ ⚂ ⚃ ⚄ ⚅
Lg. Straight Sequence of 5	SCORE 40	⚀ ⚁ ⚂ ⚃ ⚄
YAHTZEE 5 of a kind	SCORE 50	⚂ ⚂ ⚂ ⚂ ⚂
Chance	Score Total Of All 5 Dice	⚀ ⚀ ⚂ ⚂ ⚄

yahtzee_tests.asm

```
%include "mylib.inc"






section .text
global start
start:







        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

yahtzee_tests.asm

```
%include "mylib.inc"







section .text
global start
start:







        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

yahtzee_tests.asm

```
%include "mylib.inc"





section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

yahtzee_tests.asm

```
%include "mylib.inc"





section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

yahtzee_tests.asm

```
%include "mylib.inc"



section .data
dice_11122 dd 1,1,1,2,2

section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

yahtzee_tests.asm

```
%include "mylib.inc"



section .data
dice_11122 dd 1,1,1,2,2

section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

yahtzee_tests.asm

```
%include "mylib.inc"

extern score_three_of_a_kind

section .data
dice_11122 dd 1,1,1,2,2

section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

yahtzee_tests.asm

```
%include "mylib.inc"

extern score_three_of_a_kind

section .data
dice_11122 dd 1,1,1,2,2

section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

yahtzee.asm

yahtzee_tests.asm

```
%include "mylib.inc"

extern score_three_of_a_kind

section .data
dice_11122 dd 1,1,1,2,2

section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

yahtzee.asm

```
global score_three_of_a_kind
score_three_of_a_kind:
        mov eax, dword ??
        ret
```

yahtzee_tests.asm

```
%include "mylib.inc"

extern score_three_of_a_kind

section .data
dice_11122 dd 1,1,1,2,2

section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

yahtzee.asm

```
global score_three_of_a_kind
score_three_of_a_kind:
        mov eax, dword ??
        ret
```

what should we return?

yahtzee_tests.asm

```
%include "mylib.inc"

extern score_three_of_a_kind

section .data
dice_11122 dd 1,1,1,2,2

section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

yahtzee.asm

```
global score_three_of_a_kind
score_three_of_a_kind:
        mov eax, dword ??
        ret
```

what should we return?

Remember fail-fix-pass? Return something that you know will fail.

yahtzee_tests.asm

```
%include "mylib.inc"

extern score_three_of_a_kind

section .data
dice_11122 dd 1,1,1,2,2

section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

yahtzee.asm

```
global score_three_of_a_kind
score_three_of_a_kind:
        mov eax, dword 0
        ret
```

yahtzee_tests.asm

```
%include "mylib.inc"

extern score_three_of_a_kind

section .data
dice_11122 dd 1,1,1,2,2

section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

yahtzee.asm

```
global score_three_of_a_kind
score_three_of_a_kind:
        mov eax, dword 0
        ret
```

yahtzee_tests.asm

```
%include "mylib.inc"

extern score_three_of_a_kind

section .data
dice_11122 dd 1,1,1,2,2

section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

```
make: *** [check] Error 1
```

yahtzee.asm

```
global score_three_of_a_kind
score_three_of_a_kind:
        mov eax, dword 0
        ret
```

yahtzee_tests.asm

```
%include "mylib.inc"

extern score_three_of_a_kind

section .data
dice_11122 dd 1,1,1,2,2

section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

Fail - Fix - Pass

```
make: *** [check] Error 1
```

yahtzee.asm

```
global score_three_of_a_kind
score_three_of_a_kind:
        mov eax, dword 0
        ret
```

yahtzee_tests.asm

```
%include "mylib.inc"

extern score_three_of_a_kind

section .data
dice_11122 dd 1,1,1,2,2

section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

Fail - Fix - Pass

```
make: *** [check] Error 1
```

yahtzee.asm

```
global score_three_of_a_kind
score_three_of_a_kind:
        mov eax, dword 7
        ret
```

yahtzee_tests.asm

```
%include "mylib.inc"

extern score_three_of_a_kind

section .data
dice_11122 dd 1,1,1,2,2

section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

yahtzee.asm

```
global score_three_of_a_kind
score_three_of_a_kind:
        mov eax, dword 7
        ret
```

yahtzee_tests.asm

```
%include "mylib.inc"

extern score_three_of_a_kind

section .data
dice_11122 dd 1,1,1,2,2

section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

```
./yahtzee_tests
```

yahtzee.asm

```
global score_three_of_a_kind
score_three_of_a_kind:
        mov eax, dword 7
        ret
```

Fail - **Fix** - Pass

yahtzee_tests.asm

```
%include "mylib.inc"

extern score_three_of_a_kind

section .data
dice_11122 dd 1,1,1,2,2

section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

```
./yahtzee_tests
```

yahtzee.asm

```
global score_three_of_a_kind
score_three_of_a_kind:
        mov eax, dword 7
        ret
```

Fail - **Fix** - Pass

yahtzee_tests.asm

```
%include "mylib.inc"

extern score_three_of_a_kind

section .data
dice_11122 dd 1,1,1,2,2

section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

Fail - Fix - **Pass**

```
./yahtzee_tests
```

yahtzee.asm

```
global score_three_of_a_kind
score_three_of_a_kind:
        mov eax, dword 7
        ret
```

Fail - **Fix** - Pass

Congratulation. We have completed our first proper fail-fix-pass cycle.

Returning 7 is a minimal change to make it pass. This is OK because what we are concerned about now is just to make sure that the "wiring" of the test is OK. Is the test really being called and is it testing the right function?

yahtzee_tests.asm

```
%include "mylib.inc"

extern score_three_of_a_kind

section .data
dice_11122 dd 1,1,1,2,2

section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

Fail - Fix - **Pass**

```
./yahtzee_tests
```

yahtzee.asm

```
global score_three_of_a_kind
score_three_of_a_kind:
        mov eax, dword 7
        ret
```

Fail - **Fix** - Pass

Congratulation. We have completed our first proper fail-fix-pass cycle.

Returning 7 is a minimal change to make it pass. This is OK because what we are concerned about now is just to make sure that the “wiring” of the test is OK. Is the test really being called and is it testing the right function?

yahtzee_tests.asm

```
%include "mylib.inc"

extern score_three_of_a_kind

section .data
dice_11122 dd 1,1,1,2,2

section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

Let’s add another unit test.

Fail - Fix - **Pass**

```
./yahtzee_tests
```

yahtzee.asm

```
global score_three_of_a_kind
score_three_of_a_kind:
        mov eax, dword 7
        ret
```

Fail - **Fix** - Pass

Congratulation. We have completed our first proper fail-fix-pass cycle.

Returning 7 is a minimal change to make it pass. This is OK because what we are concerned about now is just to make sure that the "wiring" of the test is OK. Is the test really being called and is it testing the right function?

yahtzee_tests.asm

```
%include "mylib.inc"

extern score_three_of_a_kind

section .data
dice_11122 dd 1,1,1,2,2

section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

Let's add another unit test.

Fail - Fix - **Pass**

```
./yahtzee_tests
```

yahtzee_tests.asm

```
%include "mylib.inc"

extern score_three_of_a_kind

section .data
dice_11122 dd 1,1,1,2,2

section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

yahtzee_tests.asm

```
%include "mylib.inc"

extern score_three_of_a_kind

section .data
dice_11122 dd 1,1,1,2,2

section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

yahtzee_tests.asm

```
%include "mylib.inc"

extern score_three_of_a_kind

section .data
dice_11122 dd 1,1,1,2,2


section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure






        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

yahtzee_tests.asm

```
%include "mylib.inc"

extern score_three_of_a_kind

section .data
dice_11122 dd 1,1,1,2,2



section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure







        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

yahtzee_tests.asm

```
%include "mylib.inc"

extern score_three_of_a_kind

section .data
dice_11122 dd 1,1,1,2,2
dice_11134 dd 1,1,1,3,4

section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure








        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

yahtzee_tests.asm

```
%include "mylib.inc"

extern score_three_of_a_kind

section .data
dice_11122 dd 1,1,1,2,2
dice_11134 dd 1,1,1,3,4

section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        mov esi, dice_11134
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

yahtzee.asm

```
global score_three_of_a_kind
score_three_of_a_kind:
        mov eax, dword 7
        ret
```

yahtzee_tests.asm

```
%include "mylib.inc"

extern score_three_of_a_kind

section .data
dice_11122 dd 1,1,1,2,2
dice_11134 dd 1,1,1,3,4

section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        mov esi, dice_11134
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

yahtzee.asm

```
global score_three_of_a_kind
score_three_of_a_kind:
        mov eax, dword 7
        ret
```

yahtzee_tests.asm

```
%include "mylib.inc"

extern score_three_of_a_kind

section .data
dice_11122 dd 1,1,1,2,2
dice_11134 dd 1,1,1,3,4

section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        mov esi, dice_11134
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

```
make: *** [check] Error 1
```

yahtzee.asm

```
global score_three_of_a_kind
score_three_of_a_kind:
        mov eax, dword 7
        ret
```

yahtzee_tests.asm

```
%include "mylib.inc"

extern score_three_of_a_kind

section .data
dice_11122 dd 1,1,1,2,2
dice_11134 dd 1,1,1,3,4

section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        mov esi, dice_11134
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

Fail - Fix - Pass

```
make: *** [check] Error 1
```

yahtzee.asm

```
global score_three_of_a_kind
score_three_of_a_kind:
        mov eax, dword 7
        ret
```

Should we "cheat" again and check for the last dice, if 4 then return 10 otherwise 7?

yahtzee_tests.asm

```
%include "mylib.inc"

extern score_three_of_a_kind

dice_11122 dd 1,1,1,2,2
dice_11134 dd 1,1,1,3,4

section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        mov esi, dice_11134
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

Fail - Fix - Pass

```
make: *** [check] Error 1
```

yahtzee.asm

```
global score_three_of_a_kind
score_three_of_a_kind:
        mov eax, dword 7
        ret
```

yahtzee_tests.asm

```
%include "mylib.inc"

extern score_three_of_a_kind

dice_11122 dd 1,1,1,2,2

start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        mov esi, dice_11134
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

Should we "cheat" again and check for the last dice, if 4 then return 10 otherwise 7?

No! Another principle of TDD is that while you are supposed to do simple and "naive" increments you are not allowed to do "obviously stupid" stuff.

Fail - Fix - Pass

```
make: *** [check] Error 1
```

yahtzee.asm

```
global score_three_of_a_kind
score_three_of_a_kind:
        mov eax, dword 7
        ret
```

yahtzee_tests.asm

```
%include "mylib.inc"

extern score_three_of_a_kind
```

```
dice_11122 dd 1,1,1,2,2
```

```
start:
```

```
_a_kind
```

```
ure
```

```
        mov esi, dice_11134
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

Should we "cheat" again and check for the last dice, if 4 then return 10 otherwise 7?

No! Another principle of TDD is that while you are supposed to do simple and "naive" increments you are not allowed to do "obviously stupid" stuff.

A simple and naive thing to do here is to just **sum the dice** and return the value. That would satisfy all the tests and we know the functionality will eventually be needed.

Fail - Fix - Pass

```
make: *** [check] Error 1
```

yahtzee.asm

```
global score_three_of_a_kind
score_three_of_a_kind:
        mov eax, dword 7
        ret
```

yahtzee_tests.asm

```
%include "mylib.inc"

extern score_three_of_a_kind

dice_11122 dd 1,1,1,2,2

start:
                                    _a_kind

                                    lure

        mov esi, dice_11134
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

Should we “cheat” again and check for the last dice, if 4 then return 10 otherwise 7?

No! Another principle of TDD is that while you are supposed to do simple and “naive” increments you are not allowed to do “obviously stupid” stuff.

A simple and naive thing to do here is to just **sum the dice** and return the value. That would satisfy all the tests and we know the functionality will eventually be needed.

Fail - Fix - Pass

```
make: *** [check] Error 1
```

yahtzee.asm

```
global score_three_of_a_kind
score_three_of_a_kind:
        call sum_of_dice
        ret
```

yahtzee_tests.asm

```
%include "mylib.inc"

extern score_three_of_a_kind

section .data
dice_11122 dd 1,1,1,2,2
dice_11134 dd 1,1,1,3,4

section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        mov esi, dice_11134
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

yahtzee.asm

```
global score_three_of_a_kind
score_three_of_a_kind:
        call sum_of_dice
        ret
```

yahtzee_tests.asm

```
%include "mylib.inc"

extern score_three_of_a_kind

section .data
dice_11122 dd 1,1,1,2,2
dice_11134 dd 1,1,1,3,4

section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        mov esi, dice_11134
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

yahtzee.asm

```
global score_three_of_a_kind
score_three_of_a_kind:
        call sum_of_dice
        ret
```

yahtzee_tests.asm

```
%include "mylib.inc"

extern score_three_of_a_kind

section .data
dice_11122 dd 1,1,1,2,2
dice_11134 dd 1,1,1,3,4

section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        mov esi, dice_11134
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

yahtzee.asm

```
global score_three_of_a_kind
score_three_of_a_kind:
        call sum_of_dice
        ret

sum_of_dice:
        mov eax, [esi+0]
        add eax, [esi+4]
        add eax, [esi+8]
        add eax, [esi+12]
        add eax, [esi+16]
        ret
```

yahtzee_tests.asm

```
%include "mylib.inc"

extern score_three_of_a_kind

section .data
dice_11122 dd 1,1,1,2,2
dice_11134 dd 1,1,1,3,4

section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        mov esi, dice_11134
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

yahtzee.asm

```
global score_three_of_a_kind
score_three_of_a_kind:
        call sum_of_dice
        ret

sum_of_dice:
        mov eax, [esi+0]
        add eax, [esi+4]
        add eax, [esi+8]
        add eax, [esi+12]
        add eax, [esi+16]
        ret
```

Fail - **Fix** - Pass

yahtzee_tests.asm

```
%include "mylib.inc"

extern score_three_of_a_kind

section .data
                1,2,2
                1,3,4

section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        mov esi, dice_11134
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

yahtzee.asm

```
global score_three_of_a_kind
score_three_of_a_kind:
        call sum_of_dice
        ret

sum_of_dice:
        mov eax, [esi+0]
        add eax, [esi+4]
        add eax, [esi+8]
        add eax, [esi+12]
        add eax, [esi+16]
        ret
```

Fail - **Fix** - Pass

yahtzee_tests.asm

```
%include "mylib.inc"

extern score_three_of_a_kind

section .data
                 ,1,2,2
                  1,3,4

section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        mov esi, dice_11134
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

Fail - Fix - **Pass**

```
./yahtzee_tests
```

yahtzee.asm

```
global score_three_of_a_kind
score_three_of_a_kind:
        call sum_of_dice
        ret

sum_of_dice:
        mov eax, [esi+0]
        add eax, [esi+4]
        add eax, [esi+8]
        add eax, [esi+12]
        add eax, [esi+16]
        ret
```

yahtzee_tests.asm

```
...

section .data
dice_11122 dd 1,1,1,2,2
dice_11134 dd 1,1,1,3,4


section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        mov esi, dice_11134
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure






        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

yahtzee.asm

```
global score_three_of_a_kind
score_three_of_a_kind:
        call sum_of_dice
        ret

sum_of_dice:
        mov eax, [esi+0]
        add eax, [esi+4]
        add eax, [esi+8]
        add eax, [esi+12]
        add eax, [esi+16]
        ret
```

yahtzee_tests.asm

```
...

section .data
dice_11122 dd 1,1,1,2,2
dice_11134 dd 1,1,1,3,4


section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        mov esi, dice_11134
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure







        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

yahtzee.asm

```
global score_three_of_a_kind
score_three_of_a_kind:
        call sum_of_dice
        ret

sum_of_dice:
        mov eax, [esi+0]
        add eax, [esi+4]
        add eax, [esi+8]
        add eax, [esi+12]
        add eax, [esi+16]
        ret
```

yahtzee_tests.asm

```
...

section .data
dice_11122 dd 1,1,1,2,2
dice_11134 dd 1,1,1,3,4
dice_12345 dd 1,2,3,4,5

section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        mov esi, dice_11134
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure





        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

yahtzee.asm

```
global score_three_of_a_kind
score_three_of_a_kind:
        call sum_of_dice
        ret

sum_of_dice:
        mov eax, [esi+0]
        add eax, [esi+4]
        add eax, [esi+8]
        add eax, [esi+12]
        add eax, [esi+16]
        ret
```

yahtzee_tests.asm

```
...

section .data
dice_11122 dd 1,1,1,2,2
dice_11134 dd 1,1,1,3,4
dice_12345 dd 1,2,3,4,5

section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        mov esi, dice_11134
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure




        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

yahtzee.asm

```
global score_three_of_a_kind
score_three_of_a_kind:
        call sum_of_dice
        ret

sum_of_dice:
        mov eax, [esi+0]
        add eax, [esi+4]
        add eax, [esi+8]
        add eax, [esi+12]
        add eax, [esi+16]
        ret
```

yahtzee_tests.asm

```
...

section .data
dice_11122 dd 1,1,1,2,2
dice_11134 dd 1,1,1,3,4
dice_12345 dd 1,2,3,4,5

section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        mov esi, dice_11134
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure

        mov esi, dice_12345
        call score_three_of_a_kind
        cmp eax, dword 0
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

yahtzee.asm

```
global score_three_of_a_kind
score_three_of_a_kind:
        call sum_of_dice
        ret

sum_of_dice:
        mov eax, [esi+0]
        add eax, [esi+4]
        add eax, [esi+8]
        add eax, [esi+12]
        add eax, [esi+16]
        ret
```

yahtzee_tests.asm

```
...

section .data
dice_11122 dd 1,1,1,2,2
dice_11134 dd 1,1,1,3,4
dice_12345 dd 1,2,3,4,5

section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        mov esi, dice_11134
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure

        mov esi, dice_12345
        call score_three_of_a_kind
        cmp eax, dword 0
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

```
make: *** [check] Error 1
```

yahtzee.asm

```
global score_three_of_a_kind
score_three_of_a_kind:
        call sum_of_dice
        ret

sum_of_dice:
        mov eax, [esi+0]
        add eax, [esi+4]
        add eax, [esi+8]
        add eax, [esi+12]
        add eax, [esi+16]
        ret
```

yahtzee_tests.asm

```
...

section .data
dice_11122 dd 1,1,1,2,2
dice_11134 dd 1,1,1,3,4
dice_12345 dd 1,2,3,4,5

section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        mov esi, dice_11134
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure

        mov esi, dice_12345
        call score_three_of_a_kind
        cmp eax, dword 0
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

yahtzee.asm

```
global score_three_of_a_kind
score_three_of_a_kind:
        call sum_of_dice
        ret

sum_of_dice:
        mov eax, [esi+0]
        add eax, [esi+4]
        add eax, [esi+8]
        add eax, [esi+12]
        add eax, [esi+16]
        ret
```

yahtzee_tests.asm

```
...

section .data
dice_11122 dd 1,1,1,2,2
dice_11134 dd 1,1,1,3,4
dice_12345 dd 1,2,3,4,5

section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        mov esi, dice_11134
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure

        mov esi, dice_12345
        call score_three_of_a_kind
        cmp eax, dword 0
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

yahtzee.asm

```
%define TRUE 1
%define FALSE 0

global score_three_of_a_kind
score_three_of_a_kind:
        call have_3_of_a_kind
        cmp eax, TRUE
        je .return_sum
.return_zero:
        mov eax, 0
        ret
.return_sum:
        call sum_of_dice
        ret

...
```

yahtzee_tests.asm

```
...

section .data
dice_11122 dd 1,1,1,2,2
dice_11134 dd 1,1,1,3,4
dice_12345 dd 1,2,3,4,5

section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        mov esi, dice_11134
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure

        mov esi, dice_12345
        call score_three_of_a_kind
        cmp eax, dword 0
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

yahtzee.asm

```
%define TRUE 1
%define FALSE 0

global score_three_of_a_kind
score_three_of_a_kind:
        call have_3_of_a_kind
        cmp eax, TRUE
        je .return_sum
.return_zero:
        mov eax, 0
        ret
.return_sum:
        call sum_of_dice
        ret

...
```

yahtzee_tests.asm

```
...

section .data
dice_11122 dd 1,1,1,2,2
dice_11134 dd 1,1,1,3,4
dice_12345 dd 1,2,3,4,5

section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        mov esi, dice_11134
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure

        mov esi, dice_12345
        call score_three_of_a_kind
        cmp eax, dword 0
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

yahtzee.asm

```
%define TRUE 1
%define FALSE 0

global score_three_of_a_kind
score_three_of_a_kind:
        call have_3_of_a_kind
        cmp eax, TRUE
        je .return_sum
.return_zero:
        mov eax, 0
        ret
.return_sum:
        call sum_of_dice
        ret

...
```

yahtzee_tests.asm

```
...

section .data
dice_11122 dd 1,1,1,2,2
dice_11134 dd 1,1,1,3,4
dice_12345 dd 1,2,3,4,5

section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        mov esi, dice_11134
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure

        mov esi, dice_12345
        call score_three_of_a_kind
        cmp eax, dword 0
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

yahtzee.asm

```
%define TRUE 1
%define FALSE 0

global score_three_of_a_kind
score_three_of_a_kind:
        call have_3_of_a_kind
        cmp eax, TRUE
        je .return_sum
.return_zero:
        mov eax, 0
        ret
.return_sum:
        call sum_of_dice
        ret

have_3_of_a_kind:
        mov ebx, 1 ; face value
.check_next_face_value:
        call count_face
        cmp eax, 3
        je .return_true
        inc ebx
        cmp ebx, 6
        jg .return_false
        jmp .check_next_face_value
.return_false:
        mov eax, FALSE
        ret
.return_true:
        mov eax, TRUE
        ret
```

yahtzee_tests.asm

```
...

section .data
dice_11122 dd 1,1,1,2,2
dice_11134 dd 1,1,1,3,4
dice_12345 dd 1,2,3,4,5

section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        mov esi, dice_11134
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure

        mov esi, dice_12345
        call score_three_of_a_kind
        cmp eax, dword 0
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

yahtzee.asm

```
%define TRUE 1
%define FALSE 0

global score_three_of_a_kind
score_three_of_a_kind:
        call have_3_of_a_kind
        cmp eax, TRUE
        je .return_sum
.return_zero:
        mov eax, 0
        ret
.return_sum:
        call sum_of_dice
        ret

have_3_of_a_kind:
        mov ebx, 1 ; face value
.check_next_face_value:
        call count_face
        cmp eax, 3
        je .return_true
        inc ebx
        cmp ebx, 6
        jg .return_false
        jmp .check_next_face_value
.return_false:
        mov eax, FALSE
        ret
.return_true:
        mov eax, TRUE
        ret
```

yahtzee_tests.asm

```
...

section .data
dice_11122 dd 1,1,1,2,2
dice_11134 dd 1,1,1,3,4
dice_12345 dd 1,2,3,4,5

section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        mov esi, dice_11134
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure

        mov esi, dice_12345
        call score_three_of_a_kind
        cmp eax, dword 0
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

yahtzee.asm

```
%define TRUE 1
%define FALSE 0

global score_three_of_a_kind
score_three_of_a_kind:
        call have_3_of_a_kind
        cmp eax, TRUE
        je .return_sum
.return_zero:
        mov eax, 0
        ret
.return_sum:
        call sum_of_dice
        ret

have_3_of_a_kind:
        mov ebx, 1 ; face value
.check_next_face_value:
        call count_face
        cmp eax, 3
        je .return_true
        inc ebx
        cmp ebx, 6
        jg .return_false
        jmp .check_next_face_value
.return_false:
        mov eax, FALSE
        ret
.return_true:
        mov eax, TRUE
        ret
```

```
count_face:
        mov eax, 0
        cmp ebx, [esi+0]
        jne .next1
        inc eax
.next1:
        cmp ebx, [esi+4]
        jne .next2
        inc eax
.next2:
        cmp ebx, [esi+8]
        jne .next3
        inc eax
.next3:
        cmp ebx, [esi+12]
        jne .next4
        inc eax
.next4:
        cmp ebx, [esi+16]
        jne .next5
        inc eax
.next5:
        ret
```

```
_a_kind

ure

_a_kind

ure

_a_kind
        cmp eax, dword 0
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

yahtzee.asm

```
%define TRUE 1
%define FALSE 0

global score_three_of_a_kind
score_three_of_a_kind:
        call have_3_of_a_kind
        cmp eax, TRUE
        je .return_sum
.return_zero:
        mov eax, 0
        ret
.return_sum:
        call sum_of_dice
        ret

have_3_of_a_kind:
        mov ebx, 1 ; face value
.check_next_face_value:
        call count_face
        cmp eax, 3
        je .return_true
        inc ebx
        cmp ebx, 6
        jg .return_false
        jmp .check_next_face_value
.return_false:
        mov eax, FALSE
        ret
.return_true:
        mov eax, TRUE
        ret
```

```
count_face:
        mov eax, 0
        cmp ebx, [esi+0]
        jne .next1
        inc eax
.next1:
        cmp ebx, [esi+4]
        jne .next2
        inc eax
.next2:
        cmp ebx, [esi+8]
        jne .next3
        inc eax
.next3:
        cmp ebx, [esi+12]
        jne .next4
        inc eax
.next4:
        cmp ebx, [esi+16]
        jne .next5
        inc eax
.next5:
        ret
```

```
_a_kind

ure

_a_kind

ure

_a_kind
        cmp eax, dword 0
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

```
./yahtzee_tests
```

yahtzee_tests.asm

```
...
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        mov esi, dice_11134
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure

        mov esi, dice_12345
        call score_three_of_a_kind
        cmp eax, dword 0
        jne .exit_with_failure
...
```

```
./yahtzee_tests
```

yahtzee_tests.asm

```
...
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        mov esi, dice_11134
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure

        mov esi, dice_12345
        call score_three_of_a_kind
        cmp eax, dword 0
        jne .exit_with_failure









...
```

```
./yahtzee_tests
```

yahtzee_tests.asm

```
...
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        mov esi, dice_11134
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure

        mov esi, dice_12345
        call score_three_of_a_kind
        cmp eax, dword 0
        jne .exit_with_failure

        mov esi, dice_53552
        call score_three_of_a_kind
        cmp eax, dword 20
        jne .exit_with_failure









...
```

yahtzee_tests.asm

```
...
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        mov esi, dice_11134
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure

        mov esi, dice_12345
        call score_three_of_a_kind
        cmp eax, dword 0
        jne .exit_with_failure

        mov esi, dice_53552
        call score_three_of_a_kind
        cmp eax, dword 20
        jne .exit_with_failure








...
```

```
./yahtzee_tests
```

yahtzee_tests.asm

```
...
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        mov esi, dice_11134
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure

        mov esi, dice_12345
        call score_three_of_a_kind
        cmp eax, dword 0
        jne .exit_with_failure

        mov esi, dice_53552
        call score_three_of_a_kind
        cmp eax, dword 20
        jne .exit_with_failure







...
```

```
./yahtzee_tests
```

yahtzee_tests.asm

```
...
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        mov esi, dice_11134
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure

        mov esi, dice_12345
        call score_three_of_a_kind
        cmp eax, dword 0
        jne .exit_with_failure

        mov esi, dice_53552
        call score_three_of_a_kind
        cmp eax, dword 20
        jne .exit_with_failure





...
```

```
./yahtzee_tests
```

yahtzee_tests.asm

```
...
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        mov esi, dice_11134
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure

        mov esi, dice_12345
        call score_three_of_a_kind
        cmp eax, dword 0
        jne .exit_with_failure

        mov esi, dice_53552
        call score_three_of_a_kind
        cmp eax, dword 20
        jne .exit_with_failure

        mov esi, dice_11666
        call score_three_of_a_kind
        cmp eax, dword 20
        jne .exit_with_failure





...
```

yahtzee_tests.asm

```
...
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        mov esi, dice_11134
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure

        mov esi, dice_12345
        call score_three_of_a_kind
        cmp eax, dword 0
        jne .exit_with_failure

        mov esi, dice_53552
        call score_three_of_a_kind
        cmp eax, dword 20
        jne .exit_with_failure

        mov esi, dice_11666
        call score_three_of_a_kind
        cmp eax, dword 20
        jne .exit_with_failure
...
```

```
./yahtzee_tests
```

yahtzee_tests.asm

```
...
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        mov esi, dice_11134
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure

        mov esi, dice_12345
        call score_three_of_a_kind
        cmp eax, dword 0
        jne .exit_with_failure

        mov esi, dice_53552
        call score_three_of_a_kind
        cmp eax, dword 20
        jne .exit_with_failure

        mov esi, dice_11666
        call score_three_of_a_kind
        cmp eax, dword 20
        jne .exit_with_failure




...
```

```
./yahtzee_tests
```

yahtzee_tests.asm

```
...
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        mov esi, dice
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure

        mov esi, dice_12345
        call score_three_of_a_kind
        cmp eax, dword 0
        jne .exit_with_failure

        mov esi, dice_53552
        call score_three_of_a_kind
        cmp eax, dword 20
        jne .exit_with_failure

        mov esi, dice_11666
        call score_three_of_a_kind
        cmp eax, dword 20
        jne .exit_with_failure




...
```

Looking good! Looking good!

```
./yahtzee_tests
```

yahtzee_tests.asm

```
...
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        mov esi, dice
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure

        mov esi, dice_12345
        call score_three_of_a_kind
        cmp eax, dword 0
        jne .exit_with_failure

        mov esi, dice_53552
        call score_three_of_a_kind
        cmp eax, dword 20
        jne .exit_with_failure

        mov esi, dice_11666
        call score_three_of_a_kind
        cmp eax, dword 20
        jne .exit_with_failure




...
```

Looking good! Looking good!

```
./yahtzee_tests
```

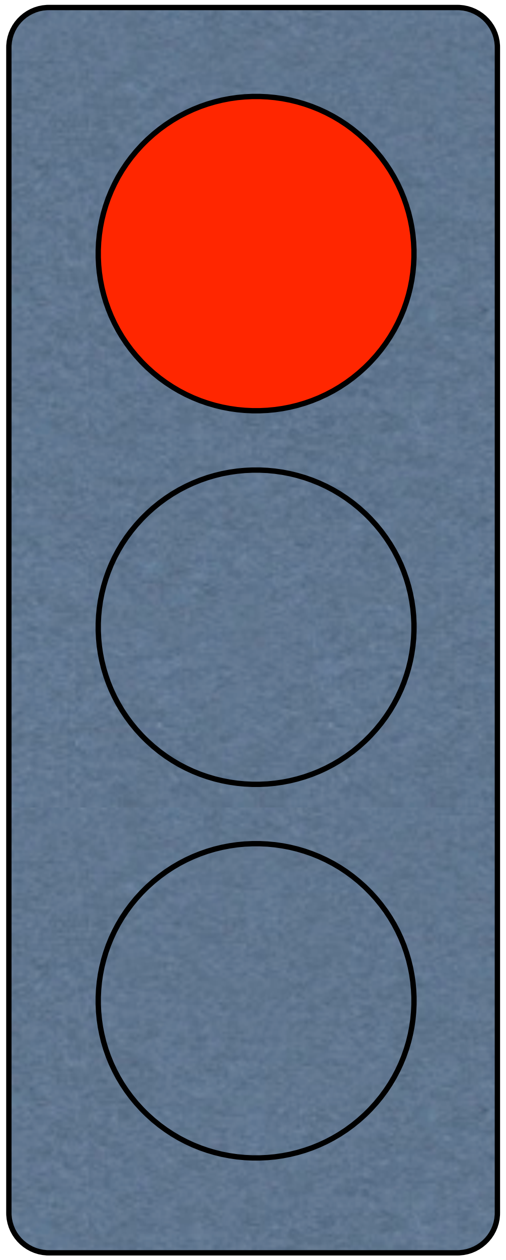
yahtzee_tests.asm

```
...
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        mov esi, dice_11134
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure

        mov esi, dice_12345
        call score_three_of_a_kind
        cmp eax, dword 0
        jne .exit_with_failure

        mov esi, dice_53552
        call score_three_of_a_kind
        cmp eax, dword 20
        jne .exit_with_failure

        mov esi, dice_11666
        call score_three_of_a_kind
        cmp eax, dword 20
        jne .exit_with_failure

        mov esi, dice_61666
        call score_three_of_a_kind
        cmp eax, dword 18+7
        jne .exit_with_failure
...
```

yahtzee_tests.asm

```
...
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        mov esi, dice_11134
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure

        mov esi, dice_12345
        call score_three_of_a_kind
        cmp eax, dword 0
        jne .exit_with_failure

        mov esi, dice_53552
        call score_three_of_a_kind
        cmp eax, dword 20
        jne .exit_with_failure

        mov esi, dice_11666
        call score_three_of_a_kind
        cmp eax, dword 20
        jne .exit_with_failure

        mov esi, dice_61666
        call score_three_of_a_kind
        cmp eax, dword 18+7
        jne .exit_with_failure
...
```

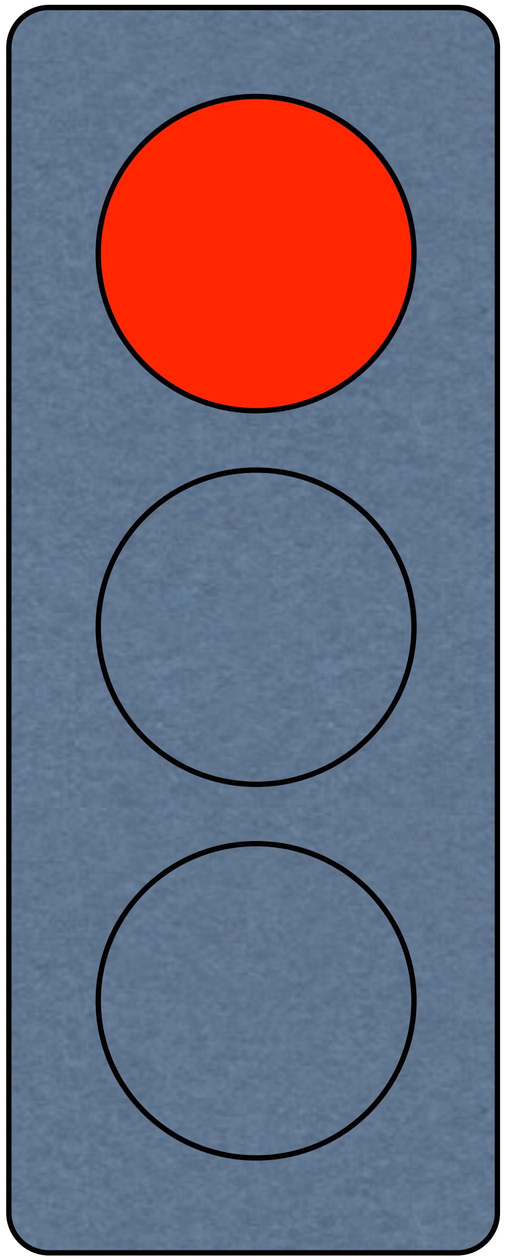
```
make: *** [check] Error 1
```



yahtzee_tests.asm

```
...
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        mov esi, dice_11134
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure

        mov esi, dice_12345
        call score_three_of_a_kind
        cmp eax, dword 0
        jne .exit_with_failure

        mov esi, dice_53552
        call score_three_of_a_kind
        cmp eax, dword 20
        jne .exit_with_failure

        mov esi, dice_11666
        call score_three_of_a_kind
        cmp eax, dword 20
        jne .exit_with_failure

        mov esi, dice_61666
        call score_three_of_a_kind
        cmp eax, dword 18+7
        jne .exit_with_failure
...
```

```
make: *** [check] Error 1
```



yahtzee_tests.asm

```
...
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        mov esi, dice_11134
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure

        mov esi, dice_12345
        call score_three_of_a_kind
        cmp eax, dword 0
        jne .exit_with_failure

        mov esi, dice_53552
        call score_three_of_a_kind
        cmp eax, dword 20
        jne .exit_with_failure

        mov esi, dice_11666
        call score_three_of_a_kind
        cmp eax, dword 20
        jne .exit_with_failure

        mov esi, dice_61666
        call score_three_of_a_kind
        cmp eax, dword 18+7
        jne .exit_with_failure
...
```

Oh no... what happened?

```
make: *** [check] Error 1
```

yahtzee.asm

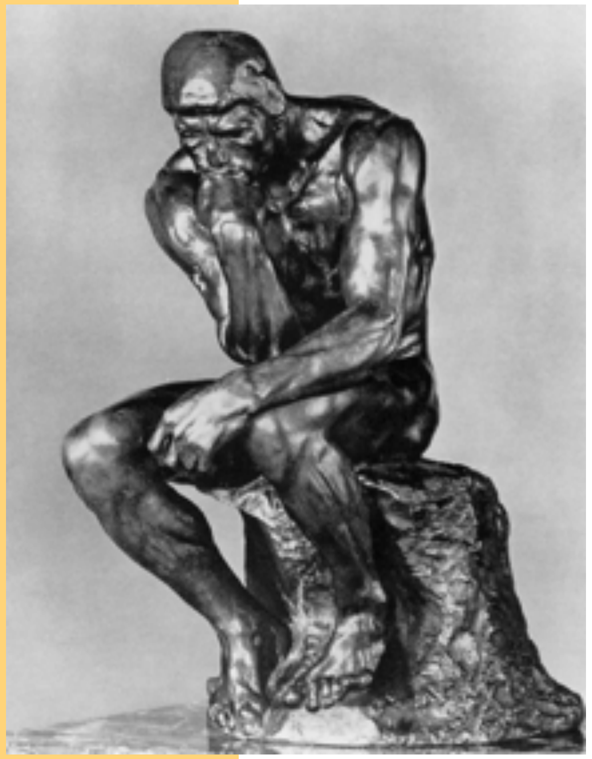
```
%define TRUE 1
%define FALSE 0

global score_three_of_a_kind
score_three_of_a_kind:
        call have_3_of_a_kind
        cmp eax, TRUE
        je .return_sum
.return_zero:
        mov eax, 0
        ret
.return_sum:
        call sum_of_dice
        ret

have_3_of_a_kind:
        mov ebx, 1 ; face value
.check_next_face_value:
        call count_face
        cmp eax, 3
        je .return_true
        inc ebx
        cmp ebx, 6
        jg .return_false
        jmp .check_next_face_value
.return_false:
        mov eax, FALSE
        ret
.return_true:
        mov eax, TRUE
        ret
```

```
count_face:
        mov eax, 0
        cmp ebx, [esi+0]
        jne .next1
        inc eax
.next1:
        cmp ebx, [esi+4]
        jne .next2
        inc eax
.next2:
        cmp ebx, [esi+8]
        jne .next3
        inc eax
.next3:
        cmp ebx, [esi+12]
        jne .next4
        inc eax
.next4:
        cmp ebx, [esi+16]
        jne .next5
        inc eax
.next5:
        ret
```

yahtzee.asm

```
%define TRUE 1
%define FALSE 0

global score_three_of_a_kind
score_three_of_a_kind:
        call have_3_of_a_kind
        cmp eax, TRUE
        je .return_sum
.return_zero:
        mov eax, 0
        ret
.return_sum:
        call sum_of_dice
        ret

have_3_of_a_kind:
        mov ebx, 1 ; face value
.check_next_face_value:
        call count_face
        cmp eax, 3
        je .return_true
        inc ebx
        cmp ebx, 6
        jg .return_false
        jmp .check_next_face_value
.return_false:
        mov eax, FALSE
        ret
.return_true:
        mov eax, TRUE
        ret
```

```
count_face:
        mov eax, 0
        cmp ebx, [esi+0]
        jne .next1
        inc eax
.next1:
        cmp ebx, [esi+4]
        jne .next2
        inc eax
.next2:
        cmp ebx, [esi+8]
        jne .next3
        inc eax
.next3:
        cmp ebx, [esi+12]
        jne .next4
        inc eax
.next4:
        cmp ebx, [esi+16]
        jne .next5
        inc eax
.next5:
        ret
```

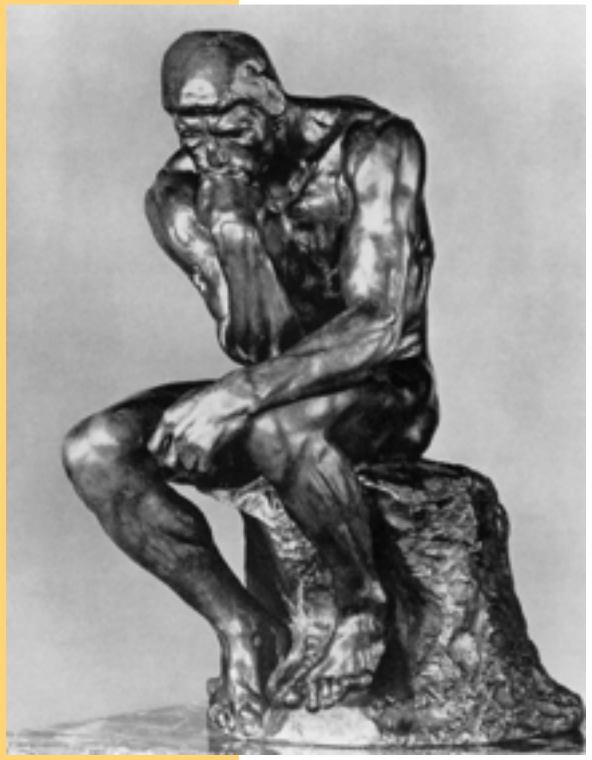


yahtzee.asm

```
%define TRUE 1
%define FALSE 0

global score_three_of_a_kind
score_three_of_a_kind:
        call have_3_of_a_kind
        cmp eax, TRUE
        je .return_sum
.return_zero:
        mov eax, 0
        ret
.return_sum:
        call sum_of_dice
        ret

have_3_of_a_kind:
        mov ebx, 1 ; face value
.check_next_face_value:
        call count_face
        cmp eax, 3
        je .return_true
        inc ebx
        cmp ebx, 6
        jg .return_false
        jmp .check_next_face_value
.return_false:
        mov eax, FALSE
        ret
.return_true:
        mov eax, TRUE
        ret
```

```
count_face:
        mov eax, 0
        cmp ebx, [esi+0]
        jne .next1
        inc eax
.next1:
        cmp ebx, [esi+4]
        jne .next2
        inc eax
.next2:
        cmp ebx, [esi+8]
        jne .next3
        inc eax
.next3:
        cmp ebx, [esi+12]
        jne .next4
        inc eax
.next4:
        cmp ebx, [esi+16]
        jne .next5
        inc eax
.next5:
        ret
```

But of course...

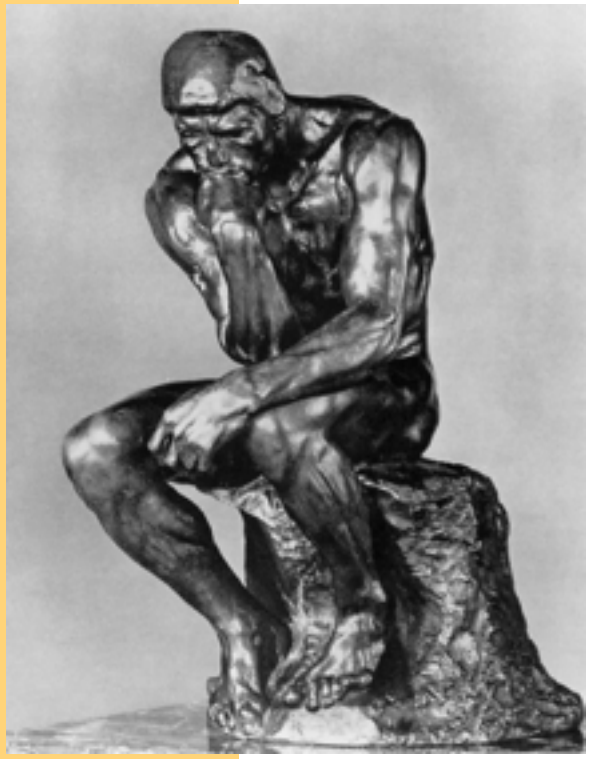


yahtzee.asm

```
%define TRUE 1
%define FALSE 0

global score_three_of_a_kind
score_three_of_a_kind:
        call have_3_of_a_kind
        cmp eax, TRUE
        je .return_sum
.return_zero:
        mov eax, 0
        ret
.return_sum:
        call sum_of_dice
        ret

have_3_of_a_kind:
        mov ebx, 1 ; face value
.check_next_face_value:
        call count_face
        cmp eax, 3
        je .return_true
        inc ebx
        cmp ebx, 6
        jg .return_false
        jmp .check_next_face_value
.return_false:
        mov eax, FALSE
        ret
.return_true:
        mov eax, TRUE
        ret
```

```
count_face:
        mov eax, 0
        cmp ebx, [esi+0]
        jne .next1
        inc eax
.next1:
        cmp ebx, [esi+4]
        jne .next2
        inc eax
.next2:
        cmp ebx, [esi+8]
        jne .next3
        inc eax
.next3:
        cmp ebx, [esi+12]
        jne .next4
        inc eax
.next4:
        cmp ebx, [esi+16]
        jne .next5
        inc eax
.next5:
        ret
```

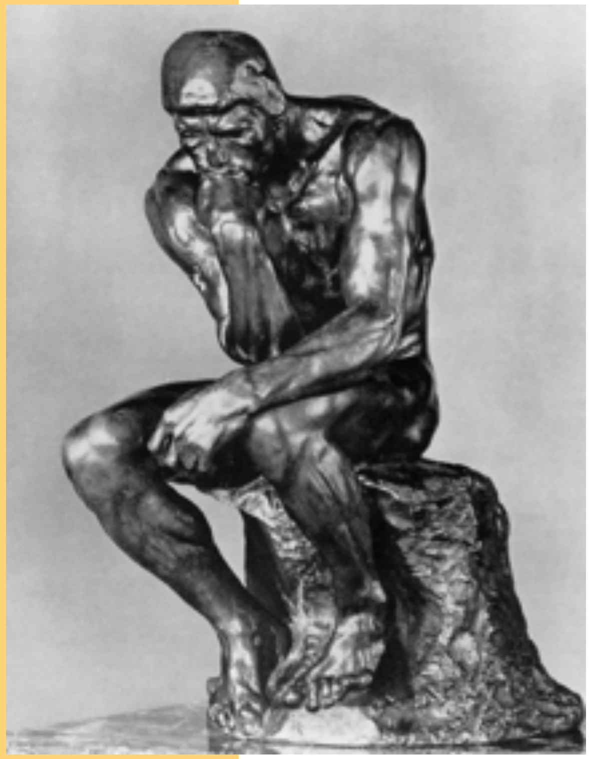
But of course...



yahtzee.asm

```
%define TRUE 1
%define FALSE 0

global score_three_of_a_kind
score_three_of_a_kind:
        call have_3_of_a_kind
        cmp eax, TRUE
        je .return_sum
.return_zero:
        mov eax, 0
        ret
.return_sum:
        call sum_of_dice
        ret

have_3_of_a_kind:
        mov ebx, 1 ; face value
.check_next_face_value:
        call count_face
        cmp eax, 3
        je .return_true
        inc ebx
        cmp ebx, 6
        jg .return_false
        jmp .check_next_face_value
.return_false:
        mov eax, FALSE
        ret
.return_true:
        mov eax, TRUE
        ret
```

```
count_face:
        mov eax, 0
        cmp ebx, [esi+0]
        jne .next1
        inc eax
.next1:
        cmp ebx, [esi+4]
        jne .next2
        inc eax
.next2:
        cmp ebx, [esi+8]
        jne .next3
        inc eax
.next3:
        cmp ebx, [esi+12]
        jne .next4
        inc eax
.next4:
        cmp ebx, [esi+16]
        jne .next5
        inc eax
.next5:
        ret
```



yahtzee.asm

```
%define TRUE 1
%define FALSE 0

global score_three_of_a_kind
score_three_of_a_kind:
        call have_3_of_a_kind
        cmp eax, TRUE
        je .return_sum
.return_zero:
        mov eax, 0
        ret
.return_sum:
        call sum_of_dice
        ret

have_3_of_a_kind:
        mov ebx, 1 ; face value
.check_next_face_value:
        call count_face
        cmp eax, 3
        je .return_true
        inc ebx
        cmp ebx, 6
        jg .return_false
        jmp .check_next_face_value
.return_false:
        mov eax, FALSE
        ret
.return_true:
        mov eax, TRUE
        ret
```

yahtzee.asm

```
%define TRUE 1
%define FALSE 0

global score_three_of_a_kind
score_three_of_a_kind:
        call have_3_of_a_kind
        cmp eax, TRUE
        je .return_sum
.return_zero:
        mov eax, 0
        ret
.return_sum:
        call sum_of_dice
        ret

have_3_of_a_kind:
        mov ebx, 1 ; face value
.check_next_face_value:
        call count_face
        cmp eax, 3
        jge .return_true
        inc ebx
        cmp ebx, 6
        jg .return_false
        jmp .check_next_face_value
.return_false:
        mov eax, FALSE
        ret
.return_true:
        mov eax, TRUE
        ret
```

yahtzee.asm

```
%define TRUE 1
%define FALSE 0

global score_three_of_a_kind
score_three_of_a_kind:
        call have_3_of_a_kind
        cmp eax, TRUE
        je .return_sum
.return_zero:
        mov eax, 0
        ret
.return_sum:
        call sum_of_dice
        ret

have_3_of_a_kind:
        mov ebx, 1 ; face value
.check_next_face_value:
        call count_face
        cmp eax, 3
        jge .return_true
        inc ebx
        cmp ebx, 6
        jg .return_false
        jmp .check_next_face_value
.return_false:
        mov eax, FALSE
        ret
.return_true:
        mov eax, TRUE
        ret
```

yahtzee.asm

```
%define TRUE 1
%define FALSE 0

global score_three_of_a_kind
score_three_of_a_kind:
        call have_at_least_3_of_a_kind
        cmp eax, TRUE
        je .return_sum
.return_zero:
        mov eax, 0
        ret
.return_sum:
        call sum_of_dice
        ret

have_at_least_3_of_a_kind:
        mov ebx, 1 ; face value
.check_next_face_value:
        call count_face
        cmp eax, 3
        jge .return_true
        inc ebx
        cmp ebx, 6
        jg .return_false
        jmp .check_next_face_value
.return_false:
        mov eax, FALSE
        ret
.return_true:
        mov eax, TRUE
        ret
```

yahtzee.asm

```
%define TRUE 1
%define FALSE 0

global score_three_of_a_kind
score_three_of_a_kind:
        call have_at_least_3_of_a_kind
        cmp eax, TRUE
        je .return_sum
.return_zero:
        mov eax, 0
        ret
.return_sum:
        call sum_of_dice
        ret

have_at_least_3_of_a_kind:
        mov ebx, 1 ; face value
.check_next_face_value:
        call count_face
        cmp eax, 3
        jge .return_true
        inc ebx
        cmp ebx, 6
        jg .return_false
        jmp .check_next_face_value
.return_false:
        mov eax, FALSE
        ret
.return_true:
        mov eax, TRUE
        ret
```

yahtzee.asm

```
%define TRUE 1
%define FALSE 0

global score_three_of_a_kind
score_three_of_a_kind:
        call have_at_least_3_of_a_kind
        cmp eax, TRUE
        je .return_sum
.return_zero:
        mov eax, 0
        ret
.return_sum:
        call sum_of_dice
        ret

have_at_least_3_of_a_kind:
        mov ebx, 1 ; face value
.check_next_face_value:
        call count_face
        cmp eax, 3
        jge .return_true
        inc ebx
        cmp ebx, 6
        jg .return_false
        jmp .check_next_face_value
.return_false:
        mov eax, FALSE
        ret
.return_true:
        mov eax, TRUE
        ret
```

```
./yahtzee_tests
```

yahtzee.asm

```
%define TRUE 1
%define FALSE 0

global score_three_of_a_kind
score_three_of_a_kind:
        call have_at_least_3_of_a_kind
        cmp eax, TRUE
        je .return_sum
.return_zero:
        mov eax, 0
        ret
.return_sum:
        call sum_of_dice
        ret

have_at_least_3_of_a_kind:
        mov ebx, 1 ; face value
.check_next_face_value:
        call count_face
        cmp eax, 3
        jge .return_true
        inc ebx
        cmp ebx, 6
        jg .return_false
        jmp .check_next_face_value
.return_false:
        mov eax, FALSE
        ret
.return_true:
        mov eax, TRUE
        ret
```

```
./yahtzee_tests
```

yahtzee.asm

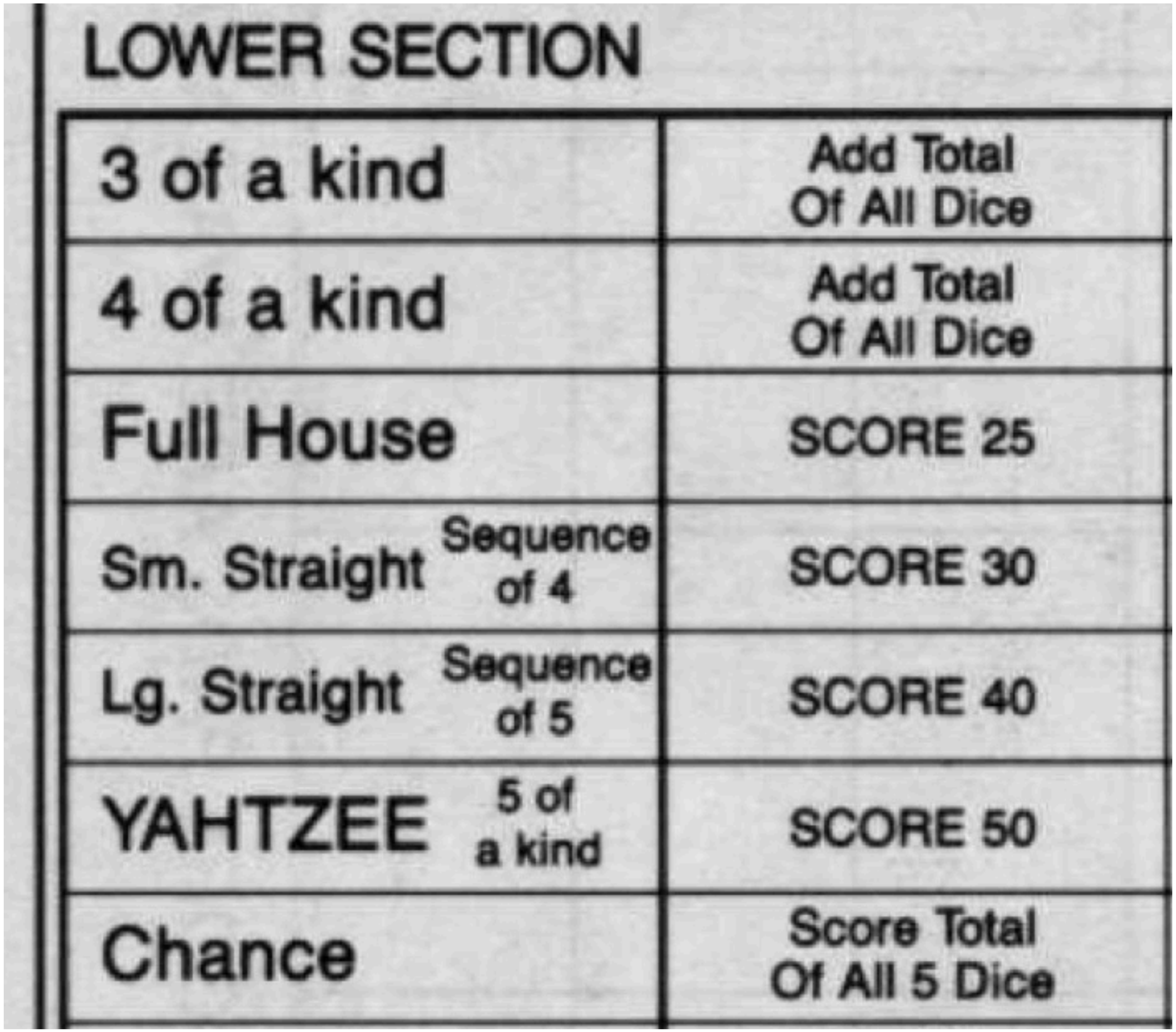
```
%define TRUE 1
%define FALSE 0

global score_three_of_a_kind
score_three_of_a_kind:
        call have_at_least_3_of_a_kind
        cmp eax, TRUE
        je .return_sum
.return_zero:
        mov eax, 0
        ret
.return_sum:
        call sum_of_dice
        ret

have_at_least_3_of_a_kind:
        mov ebx, 1 ; face value
.check_next_face_value:
        call count_face
        cmp eax, 3
        jge .return_true
        inc ebx
        cmp ebx, 6
        jg .return_false
        jmp .check_next_face_value
.return_false:
        mov eax, FALSE
        ret
.return_true:
        mov eax, TRUE
        ret
```

```
./yahtzee_tests
```

yahtzee.asm

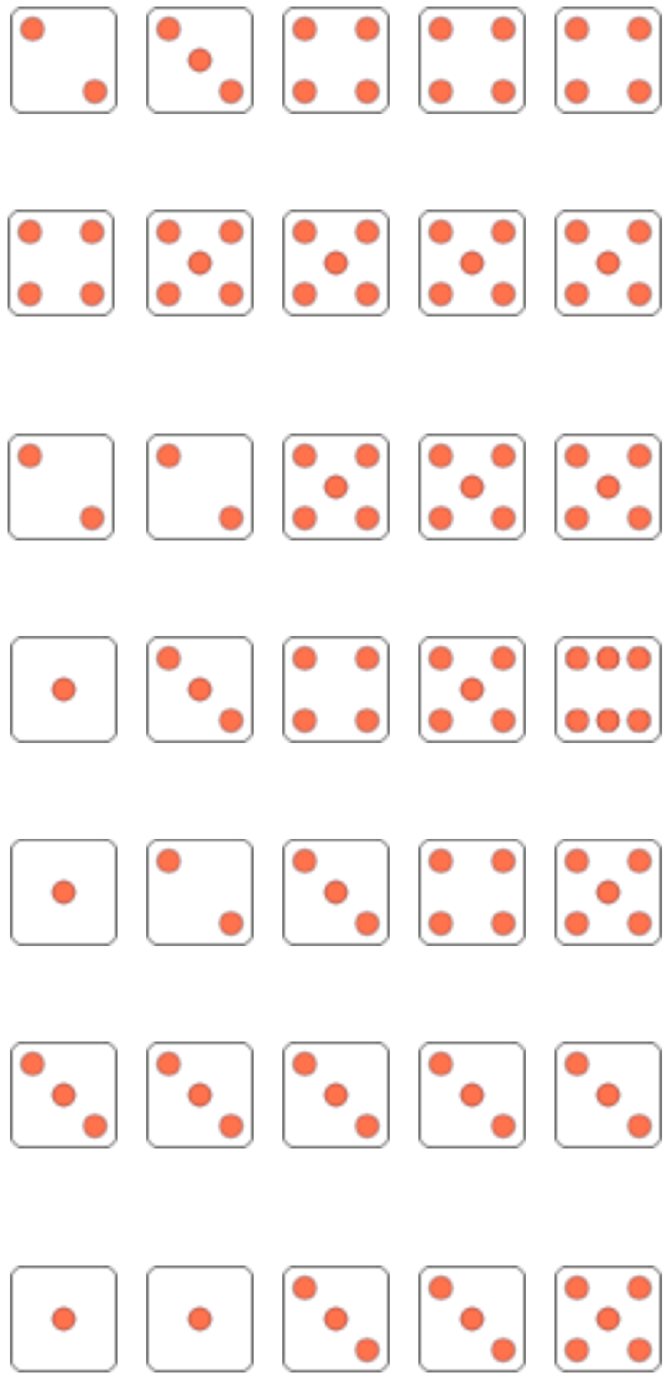
```
%define TRUE 1
%define FALSE 0

global score_three_of_a_kind
score_three_of_a_kind:
        call have_at_least_3_of_a_kind
        cmp eax, TRUE
        je .return_sum
.return_zero:
        mov eax, 0
        ret
.return_sum:
        call sum_of_dice
        ret

have_at_least_3_of_a_kind:
        mov ebx, 1 ; face value
.check_next_face_value:
        call count_face
        cmp eax, 3
        jge .return_true
        inc ebx
        cmp ebx, 6
        jg .return_false
        jmp .check_next_face_value
.return_false:
        mov eax, FALSE
        ret
.return_true:
        mov eax, TRUE
        ret
```

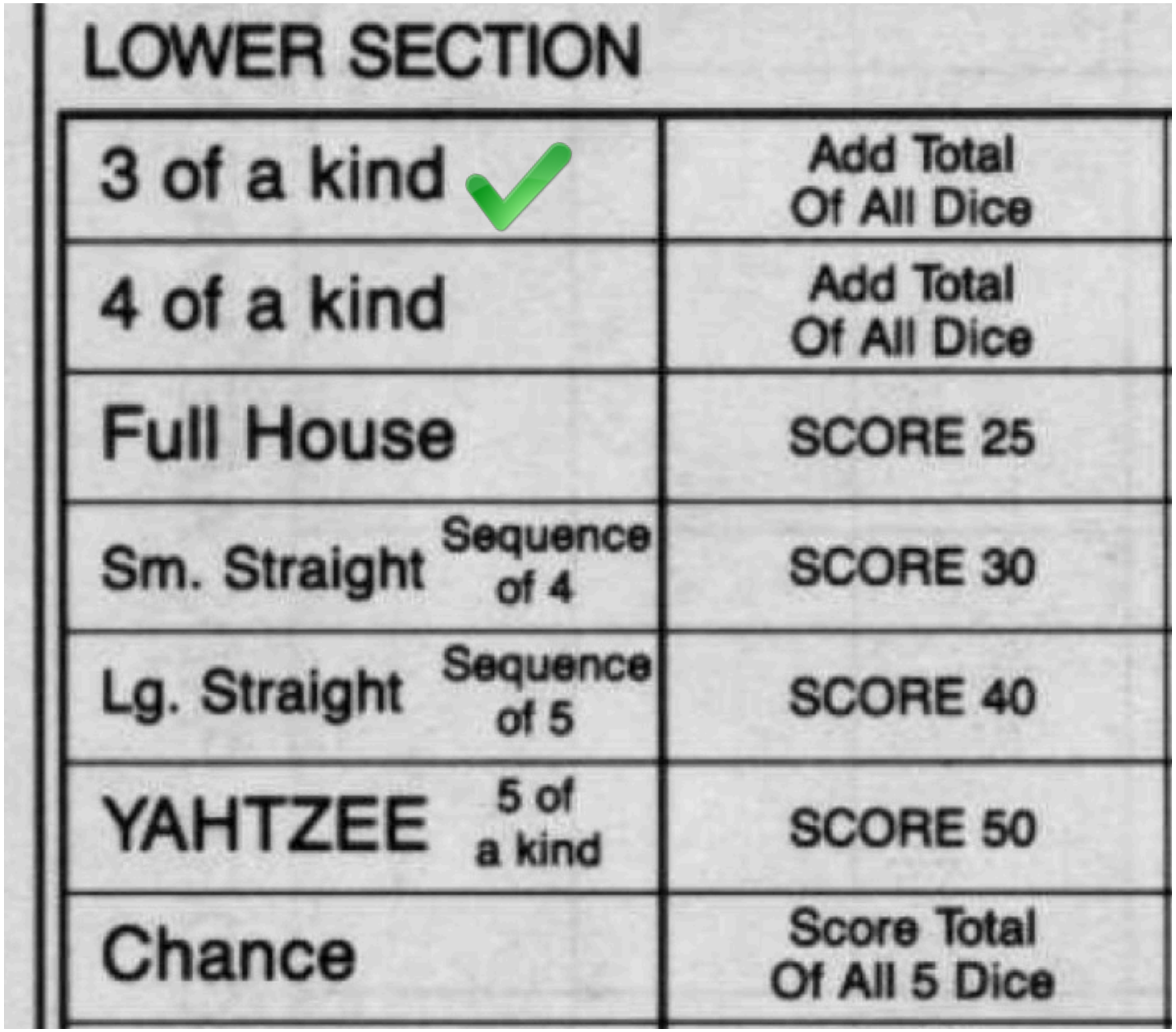
```
./yahtzee_tests
```



LOWER SECTION

3 of a kind		Add Total Of All Dice
4 of a kind		Add Total Of All Dice
Full House		SCORE 25
Sm. Straight	Sequence of 4	SCORE 30
Lg. Straight	Sequence of 5	SCORE 40
YAHTZEE	5 of a kind	SCORE 50
Chance		Score Total Of All 5 Dice





LOWER SECTION

3 of a kind ✓		Add Total Of All Dice	⚁ ⚂ ⚃ ⚃ ⚃
4 of a kind		Add Total Of All Dice	⚃ ⚄ ⚄ ⚄ ⚄
Full House		SCORE 25	⚁ ⚁ ⚄ ⚄ ⚄
Sm. Straight	Sequence of 4	SCORE 30	⚀ ⚂ ⚃ ⚄ ⚅
Lg. Straight	Sequence of 5	SCORE 40	⚀ ⚁ ⚂ ⚃ ⚄
YAHTZEE	5 of a kind	SCORE 50	⚂ ⚂ ⚂ ⚂ ⚂
Chance		Score Total Of All 5 Dice	⚀ ⚀ ⚂ ⚂ ⚄

code developed so far

```
%define TRUE 1
%define FALSE 0

global score_three_of_a_kind
score_three_of_a_kind:
        call have_at_least_3_of_a_kind
        cmp eax, TRUE
        je .return_sum
.return_zero:
        mov eax, 0
        ret
.return_sum:
        call sum_of_dice
        ret

have_at_least_3_of_a_kind:
        mov ebx, 1 ; face value
.check_next_face_value:
        call count_face
        cmp eax, 3
        jge .return_true
        inc ebx
        cmp ebx, 6
        jg .return_false
        jmp .check_next_face_value
.return_false:
        mov eax, FALSE
        ret
.return_true:
        mov eax, TRUE
        ret

count_face:
        mov eax, 0
        cmp ebx, [esi+0]
        jne .next1
        inc eax
.next1:
        cmp ebx, [esi+4]
        jne .next2
        inc eax
.next2:
        cmp ebx, [esi+8]
        jne .next3
        inc eax
.next3:
        cmp ebx, [esi+12]
        jne .next4
        inc eax
.next4:
        cmp ebx, [esi+16]
        jne .next5
        inc eax
.next5:
        ret

sum_of_dice:
        mov eax, [esi+0]
        add eax, [esi+4]
        add eax, [esi+8]
        add eax, [esi+12]
        add eax, [esi+16]
        ret
```

```
%include "mylib.inc"

extern score_three_of_a_kind

section .data
dice_11122 dd 1,1,1,2,2
dice_11134 dd 1,1,1,3,4
dice_12345 dd 1,2,3,4,5
dice_53552 dd 5,3,5,5,2
dice_11666 dd 1,1,6,6,6
dice_61666 dd 6,1,6,6,6

section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        mov esi, dice_11134
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure

        mov esi, dice_12345
        call score_three_of_a_kind
        cmp eax, dword 0
        jne .exit_with_failure

        mov esi, dice_53552
        call score_three_of_a_kind
        cmp eax, dword 20
        jne .exit_with_failure

        mov esi, dice_11666
        call score_three_of_a_kind
        cmp eax, dword 20
        jne .exit_with_failure

        mov esi, dice_61666
        call score_three_of_a_kind
        cmp eax, dword 18+7
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

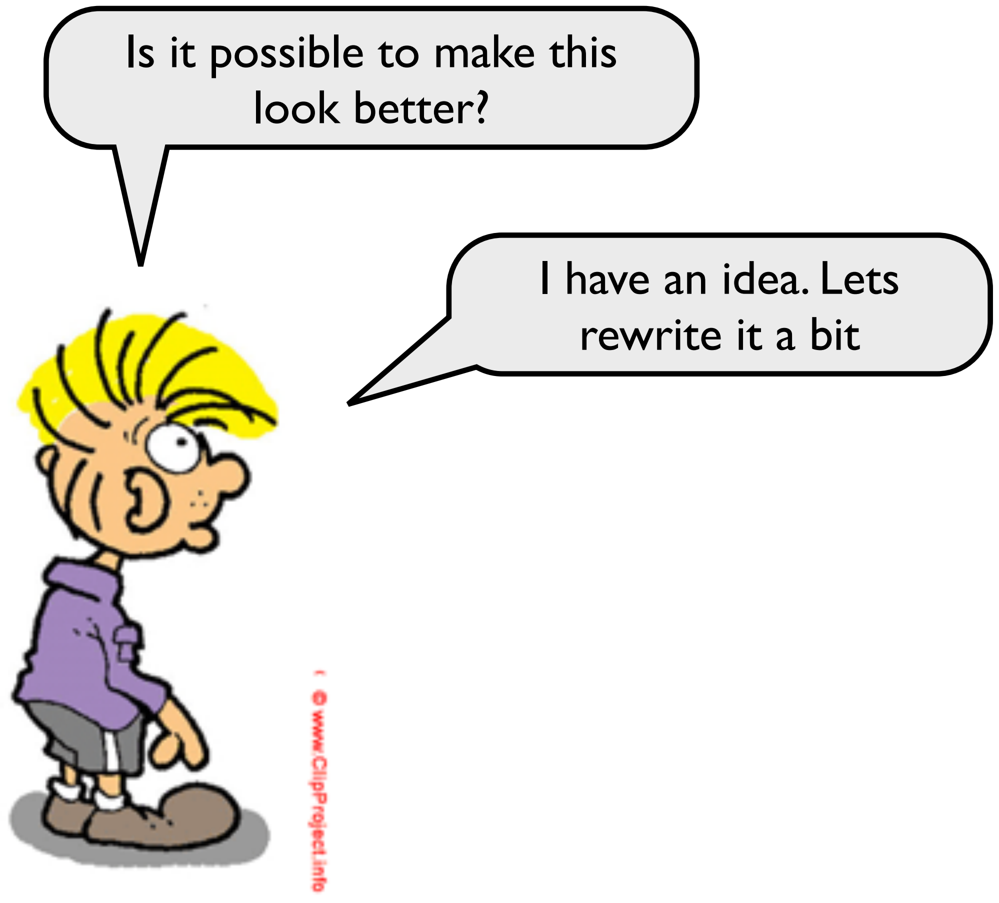
code developed so far

```
%define TRUE 1
%define FALSE 0

global score_three_of_a_kind
score_three_of_a_kind:
        call have_at_least_3_of_a_kind
        cmp eax, TRUE
        je .return_sum
.return_zero:
        mov eax, 0
        ret
.return_sum:
        call sum_of_dice
        ret

have_at_least_3_of_a_kind:
        mov ebx, 1 ; face value
.check_next_face_value:
        call count_face
        cmp eax, 3
        jge .return_true
        inc ebx
        cmp ebx, 6
        jg .return_false
        jmp .check_next_face_value
.return_false:
        mov eax, FALSE
        ret
.return_true:
        mov eax, TRUE
        ret

count_face:
        mov eax, 0
        cmp ebx, [esi+0]
        jne .next1
        inc eax
.next1:
        cmp ebx, [esi+4]
        jne .next2
        inc eax
.next2:
        cmp ebx, [esi+8]
        jne .next3
        inc eax
.next3:
        cmp ebx, [esi+12]
        jne .next4
        inc eax
.next4:
        cmp ebx, [esi+16]
        jne .next5
        inc eax
.next5:
        ret

sum_of_dice:
        mov eax, [esi+0]
        add eax, [esi+4]
        add eax, [esi+8]
        add eax, [esi+12]
        add eax, [esi+16]
        ret
```

```
%include "mylib.inc"

extern score_three_of_a_kind

section .data
dice_11122 dd 1,1,1,2,2
dice_11134 dd 1,1,1,3,4
dice_12345 dd 1,2,3,4,5
dice_53552 dd 5,3,5,5,2
dice_11666 dd 1,1,6,6,6
dice_61666 dd 6,1,6,6,6

section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        mov esi, dice_11134
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure

        mov esi, dice_12345
        call score_three_of_a_kind
        cmp eax, dword 0
        jne .exit_with_failure

        mov esi, dice_53552
        call score_three_of_a_kind
        cmp eax, dword 20
        jne .exit_with_failure

        mov esi, dice_11666
        call score_three_of_a_kind
        cmp eax, dword 20
        jne .exit_with_failure

        mov esi, dice_61666
        call score_three_of_a_kind
        cmp eax, dword 18+7
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

```
...

section .data
dice_11122 dd 1,1,1,2,2
dice_11134 dd 1,1,1,3,4
dice_12345 dd 1,2,3,4,5
dice_53552 dd 5,3,5,5,2
dice_11666 dd 1,1,6,6,6
dice_61666 dd 6,1,6,6,6

section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        mov esi, dice_11134
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure

        mov esi, dice_12345
        call score_three_of_a_kind
        cmp eax, dword 0
        jne .exit_with_failure

...
```

```
...

section .data
dice_11122 dd 1,1,1,2,2
dice_11134 dd 1,1,1,3,4
dice_12345 dd 1,2,3,4,5
dice_53552 dd 5,3,5,5,2
dice_11666 dd 1,1,6,6,6
dice_61666 dd 6,1,6,6,6

section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        mov esi, dice_11134
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure

        mov esi, dice_12345
        call score_three_of_a_kind
        cmp eax, dword 0
        jne .exit_with_failure

...
```


Is it possible to make this look better?
I have an idea. Lets rewrite it a bit
© www.ClipProject.info

```
...

section .data
dice_11122 dd 1,1,1,2,2
dice_11134 dd 1,1,1,3,4
dice_12345 dd 1,2,3,4,5
dice_53552 dd 5,3,5,5,2
dice_11666 dd 1,1,6,6,6
dice_61666 dd 6,1,6,6,6

section .text
global start
start:
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

        mov esi, dice_11134
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure

        mov esi, dice_12345
        call score_three_of_a_kind
        cmp eax, dword 0
        jne .exit_with_failure

...
```

```
...

section .text
global start
start:

section .data
dice_11122 dd 1,1,1,2,2
section .text
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

section .data
dice_11134 dd 1,1,1,3,4
section .text
        mov esi, dice_11134
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure

section .data
dice_12345 dd 1,2,3,4,5
section .text

        mov esi, dice_12345
        call score_three_of_a_kind
        cmp eax, dword 0
```

now we recognize a recurring pattern

```
...

section .text
global start
start:

section .data
dice_11122 dd 1,1,1,2,2
section .text
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

section .data
dice_11134 dd 1,1,1,3,4
section .text
        mov esi, dice_11134
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure

section .data
dice_12345 dd 1,2,3,4,5
section .text

        mov esi, dice_12345
        call score_three_of_a_kind
        cmp eax, dword 0
```

now we recognize a recurring pattern

```
...

section .text
global start
start:

section .data
dice_11122 dd 1,1,1,2,2
section .text
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

section .data
dice_11134 dd 1,1,1,3,4
section .text
        mov esi, dice_11134
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure

section .data
dice_12345 dd 1,2,3,4,5
section .text

        mov esi, dice_12345
        call score_three_of_a_kind
        cmp eax, dword 0
```

now we recognize a recurring pattern

and it is easy to introduce a macro

```
...

section .text
global start
start:

section .data
dice_11122 dd 1,1,1,2,2
section .text
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

section .data
dice_11134 dd 1,1,1,3,4
section .text
        mov esi, dice_11134
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure

section .data
dice_12345 dd 1,2,3,4,5
section .text

        mov esi, dice_12345
        call score_three_of_a_kind
        cmp eax, dword 0
```

```
%macro eval_dice 6
section .data
        %%dice dd %2,%3,%4,%5,%6
section .text
        mov esi, %%dice
        call %1
%endmacro
```

now we recognize a recurring pattern

and it is easy to introduce a macro

```
...

section .text
global start
start:

section .data
dice_11122 dd 1,1,1,2,2
section .text
        mov esi, dice_11122
        call score_three_of_a_kind
        cmp eax, dword 7
        jne .exit_with_failure

section .data
dice_11134 dd 1,1,1,3,4
section .text
        mov esi, dice_11134
        call score_three_of_a_kind
        cmp eax, dword 10
        jne .exit_with_failure

section .data
dice_12345 dd 1,2,3,4,5
section .text

        mov esi, dice_12345
        call score_three_of_a_kind
        cmp eax, dword 0
```

```
%include "mylib.inc"

%macro eval_dice 6
section .data
        %%dice dd %2,%3,%4,%5,%6
section .text
        mov esi, %%dice
        call %1
%endmacro

extern score_three_of_a_kind

section .text
global start
start:
        eval_dice score_three_of_a_kind, 1,1,1,2,2
        cmp eax, dword 7
        jne .exit_with_failure

        eval_dice score_three_of_a_kind, 1,1,1,3,4
        cmp eax, dword 10
        jne .exit_with_failure

        eval_dice score_three_of_a_kind, 1,2,3,4,5
        cmp eax, dword 0
        jne .exit_with_failure

        eval_dice score_three_of_a_kind, 5,3,5,5,2
        cmp eax, dword 20
        jne .exit_with_failure

        eval_dice score_three_of_a_kind, 1,1,6,6,6
        cmp eax, dword 20
        jne .exit_with_failure

        eval_dice score_three_of_a_kind, 6,1,6,6,6
        cmp eax, dword 18+7
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

```
%include "mylib.inc"

%macro eval_dice 6
section .data
        %%dice dd %2,%3,%4,%5,%6
section .text
        mov esi, %%dice
        call %1
%endmacro

extern score_three_of_a_kind

section .text
global start
start:
        eval_dice score_three_of_a_kind, 1,1,1,2,2
        cmp eax, dword 7
        jne .exit_with_failure

        eval_dice score_three_of_a_kind, 1,1,1,3,4
        cmp eax, dword 10
        jne .exit_with_failure

        eval_dice score_three_of_a_kind, 1,2,3,4,5
        cmp eax, dword 0
        jne .exit_with_failure

        eval_dice score_three_of_a_kind, 5,3,5,5,2
        cmp eax, dword 20
        jne .exit_with_failure

        eval_dice score_three_of_a_kind, 1,1,6,6,6
        cmp eax, dword 20
        jne .exit_with_failure

        eval_dice score_three_of_a_kind, 6,1,6,6,6
        cmp eax, dword 18+7
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

Slightly better. Perhaps we can introduce another macro as well?
© www.ClipProject.info

Slightly better. Perhaps we can introduce another macro as well?
© www.ClipProject.info

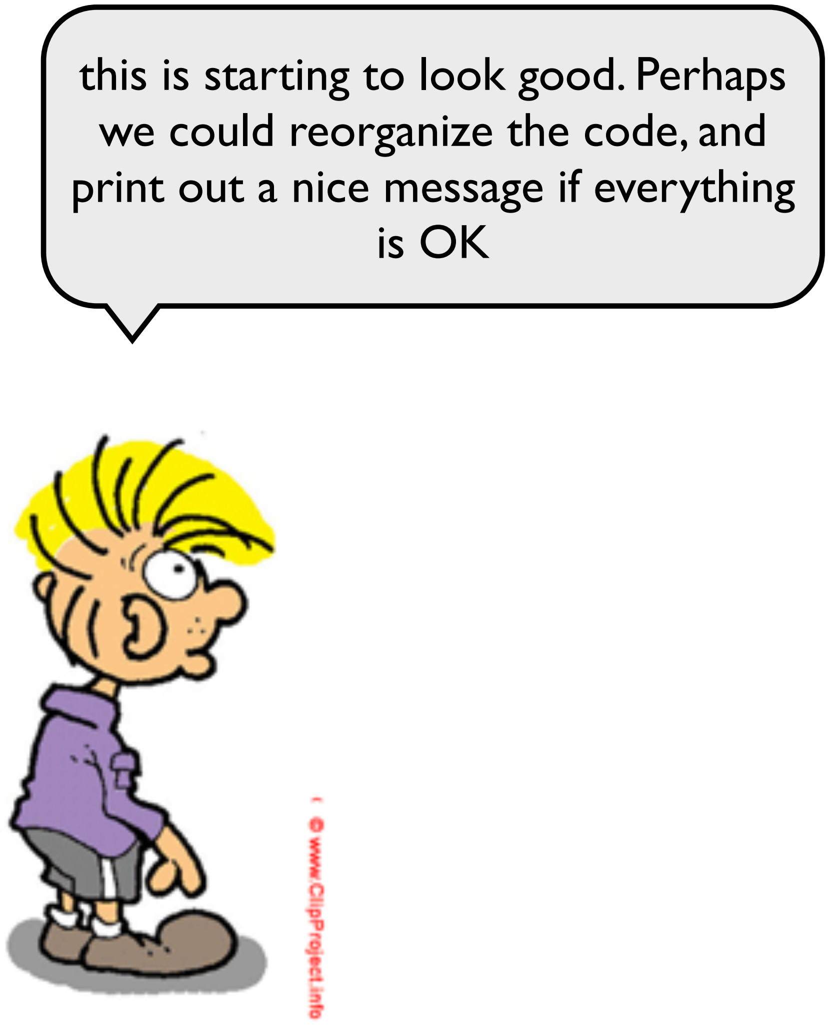
```
%include "mylib.inc"

%macro eval_dice 6
section .data
        %%dice dd %2,%3,%4,%5,%6
section .text
        mov esi, %%dice
        call %1
%endmacro

extern score_three_of_a_kind

section .text
global start
start:
        eval_dice score_three_of_a_kind, 1,1,1,2,2
        cmp eax, dword 7
        jne .exit_with_failure

        eval_dice score_three_of_a_kind, 1,1,1,3,4
        cmp eax, dword 10
        jne .exit_with_failure

        eval_dice score_three_of_a_kind, 1,2,3,4,5
        cmp eax, dword 0
        jne .exit_with_failure

        eval_dice score_three_of_a_kind, 5,3,5,5,2
        cmp eax, dword 20
        jne .exit_with_failure

        eval_dice score_three_of_a_kind, 1,1,6,6,6
        cmp eax, dword 20
        jne .exit_with_failure

        eval_dice score_three_of_a_kind, 6,1,6,6,6
        cmp eax, dword 18+7
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

```
%macro TEST_assert_eax_equals 1
        sub eax, dword %1
        jz %%ok
        sys_exit EXIT_FAILURE
%%ok:
        nop
%endmacro
```

```
%include "mylib.inc"

%macro eval_dice 6
section .data
        %%dice dd %2,%3,%4,%5,%6
section .text
        mov esi, %%dice
        call %1
%endmacro

extern score_three_of_a_kind

section .text
global start
start:
        eval_dice score_three_of_a_kind, 1,1,1,2,2
        cmp eax, dword 7
        jne .exit_with_failure

        eval_dice score_three_of_a_kind, 1,1,1,3,4
        cmp eax, dword 10
        jne .exit_with_failure

        eval_dice score_three_of_a_kind, 1,2,3,4,5
        cmp eax, dword 0
        jne .exit_with_failure

        eval_dice score_three_of_a_kind, 5,3,5,5,2
        cmp eax, dword 20
        jne .exit_with_failure

        eval_dice score_three_of_a_kind, 1,1,6,6,6
        cmp eax, dword 20
        jne .exit_with_failure

        eval_dice score_three_of_a_kind, 6,1,6,6,6
        cmp eax, dword 18+7
        jne .exit_with_failure

        sys_exit EXIT_SUCCESS
.exit_with_failure:
        sys_exit EXIT_FAILURE
```

Slightly better. Perhaps we can introduce another macro as well?
© www.ClipProject.info

```
%include "mylib.inc"

%macro TEST_assert_eax_equals 1
        sub eax, dword %1
        jz %%ok
        sys_exit EXIT_FAILURE
%%ok:
        nop
%endmacro

%macro eval_dice 6
section .data
        %%dice dd %2,%3,%4,%5,%6
section .text
        mov esi, %%dice
        call %1
%endmacro

extern score_three_of_a_kind

section .text
global start
start:
        eval_dice score_three_of_a_kind, 1,1,1,2,2
        TEST_assert_eax_equals 7

        eval_dice score_three_of_a_kind, 1,1,1,3,4
        TEST_assert_eax_equals 10

        eval_dice score_three_of_a_kind, 1,2,3,4,5
        TEST_assert_eax_equals 0

        eval_dice score_three_of_a_kind, 5,3,5,5,2
        TEST_assert_eax_equals 20

        eval_dice score_three_of_a_kind, 1,1,6,6,6
        TEST_assert_eax_equals 20

        eval_dice score_three_of_a_kind, 6,1,6,6,6
        TEST_assert_eax_equals 18+7
```


this is starting to look good. Perhaps we could reorganize the code, and print out a nice message if everything is OK
© www.ClipProject.info

```
%include "mylib.inc"

%macro TEST_assert_eax_equals 1
        sub eax, dword %1
        jz %%ok
        sys_exit EXIT_FAILURE
%%ok:
        nop
%endmacro

%macro eval_dice 6
section .data
        %%dice dd %2,%3,%4,%5,%6
section .text
        mov esi, %%dice
        call %1
%endmacro

extern score_three_of_a_kind

section .text
global start
start:
        eval_dice score_three_of_a_kind, 1,1,1,2,2
        TEST_assert_eax_equals 7

        eval_dice score_three_of_a_kind, 1,1,1,3,4
        TEST_assert_eax_equals 10

        eval_dice score_three_of_a_kind, 1,2,3,4,5
        TEST_assert_eax_equals 0

        eval_dice score_three_of_a_kind, 5,3,5,5,2
        TEST_assert_eax_equals 20

        eval_dice score_three_of_a_kind, 1,1,6,6,6
        TEST_assert_eax_equals 20

        eval_dice score_three_of_a_kind, 6,1,6,6,6
        TEST_assert_eax_equals 18+7
```

```
%include "mylib.inc"

%macro TEST_assert_eax_equals 1
        sub eax, dword %1
        jz %%ok
        sys_exit EXIT_FAILURE
%%ok:
        nop
%endmacro

%macro eval_dice 6
section .data
        %%dice dd %2,%3,%4,%5,%6
section .text
        mov esi, %%dice
        call %1
%endmacro
```

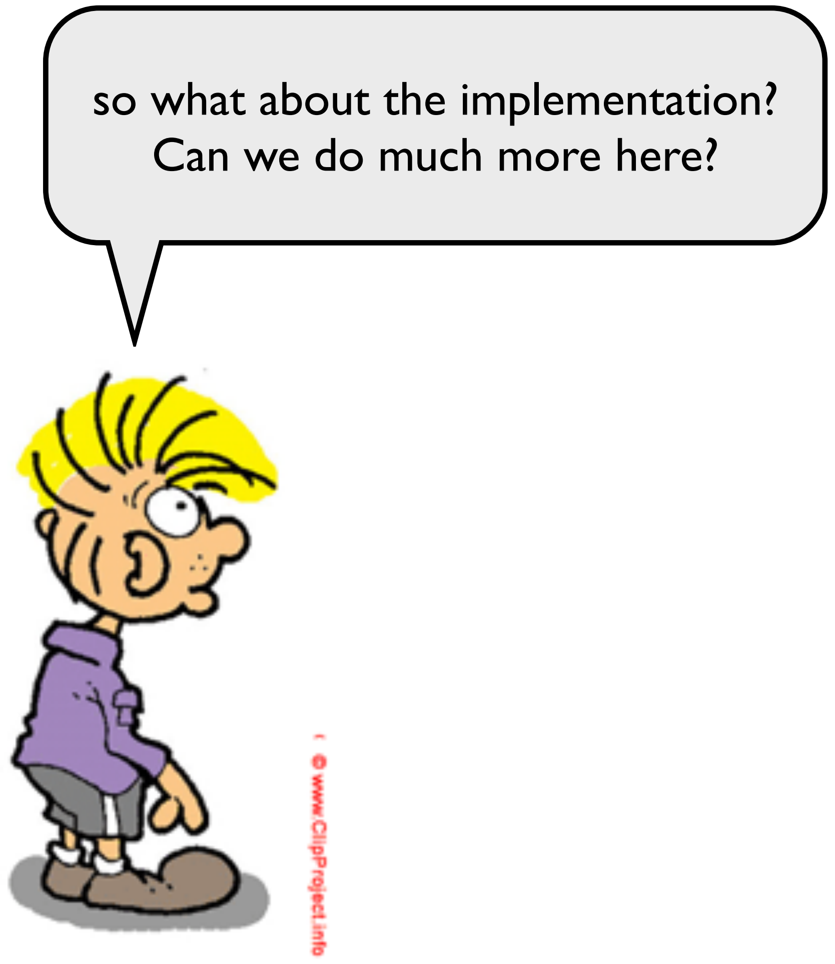
```
extern score_three_of_a_kind

check_score_three_of_a_kind:
        eval_dice score_three_of_a_kind, 1,1,1,2,2
        TEST_assert_eax_equals 7

        eval_dice score_three_of_a_kind, 1,1,1,3,4
        TEST_assert_eax_equals 10

        eval_dice score_three_of_a_kind, 1,2,3,4,5
        TEST_assert_eax_equals 0

        eval_dice score_three_of_a_kind, 5,3,5,5,2
        TEST_assert_eax_equals 20

        eval_dice score_three_of_a_kind, 1,1,6,6,6
        TEST_assert_eax_equals 20

        eval_dice score_three_of_a_kind, 6,1,6,6,6
        TEST_assert_eax_equals 18+7

        ret

section .text
global start
start:
section .data
.msg db 'Larry is invincible!', 0xa
.len equ $-.msg
section .text
        call check_score_three_of_a_kind
        sys_write STDOUT, .msg, .len
        sys_exit EXIT_SUCCESS
```

```
%include "mylib.inc"

%macro TEST_
        sub
        jz
        sys_
%%ok:
        nop
%endmacro

%macro eval_
section .da
        %%d
section .te
        mov
        cal
%endmacro
```

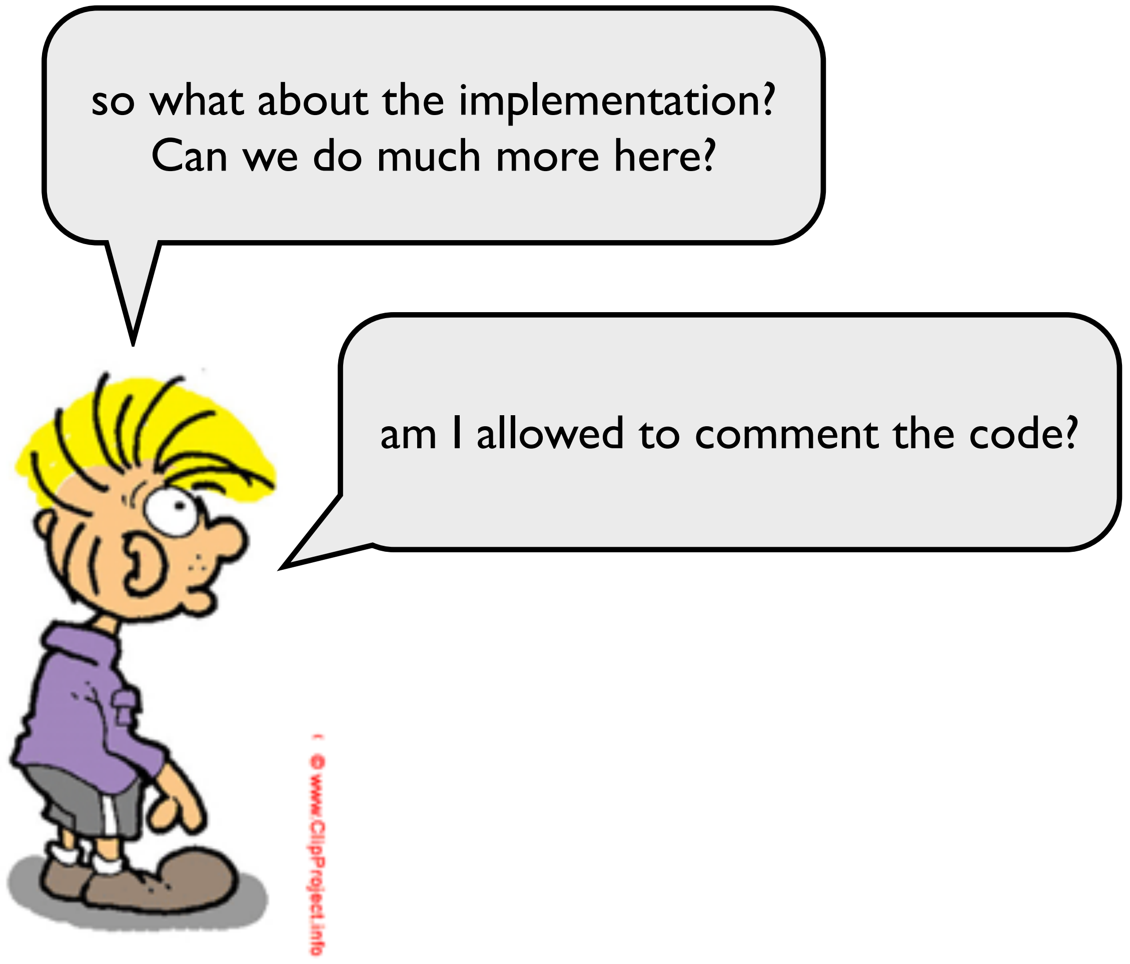
```
extern score_three_of_a_kind

nd:
hree_of_a_kind, 1,1,1,2,2
quals 7

hree_of_a_kind, 1,1,1,3,4
quals 10

hree_of_a_kind, 1,2,3,4,5
quals 0

hree_of_a_kind, 5,3,5,5,2
quals 20

hree_of_a_kind, 1,1,6,6,6
quals 20

hree_of_a_kind, 6,1,6,6,6
quals 18+7


section .text
global start
start:
section .data
.msg db 'Larry is invincible!', 0xa
.len equ $-.msg
section .text
        call check_score_three_of_a_kind
        sys_write STDOUT, .msg, .len
        sys_exit EXIT_SUCCESS
```

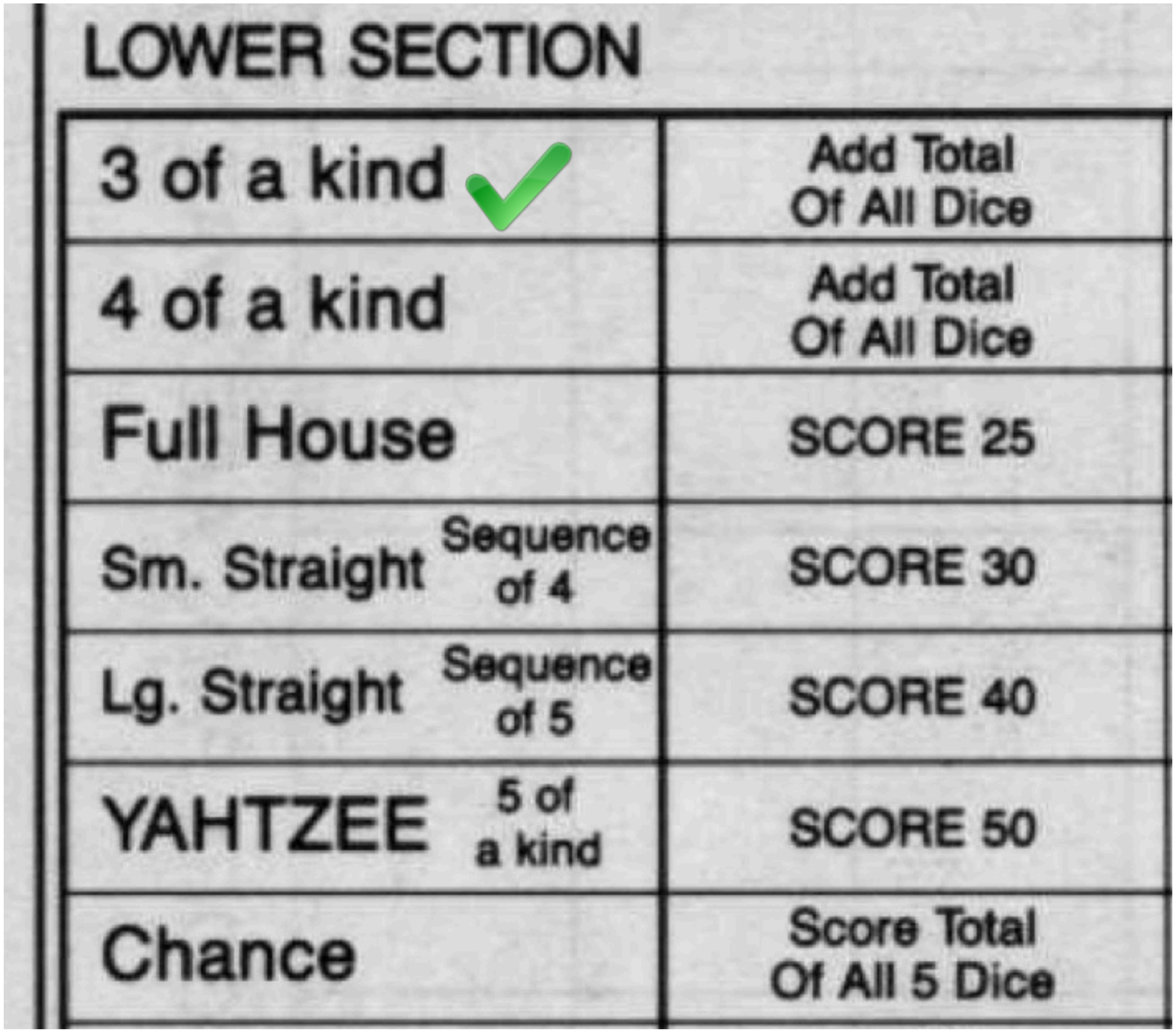
```
$ make check
nasm -f macho yahtzee_tests.asm
nasm -f macho yahtzee.asm
ar -rcs yahtzee.a yahtzee.o
ld -o yahtzee_tests yahtzee_tests.o yahtzee.a
./yahtzee_tests
Larry is invincible!
$ make check
./yahtzee_tests
Larry is invincible!
$ make check
./yahtzee_tests
Larry is invincible!
$ make check
./yahtzee_tests
Larry is invincible!
$ make check
./yahtzee_tests
Larry is invincible!
$
```

```
%define TRUE 1
%define FALSE 0

global score_three_of_a_kind
score_three_of_a_kind:
        call have_at_least_3_of_a_kind
        cmp eax, TRUE
        je .return_sum
.return_zero:
        mov eax, 0
        ret
.return_sum:
        call sum_of_dice
        ret

have_at_least_3_of_a_kind:
        mov ebx, 1 ; face value
.check_next_face_value:
        call count_face
        cmp eax, 3
        jge .return_true
        inc ebx
        cmp ebx, 6
        jg .return_false
        jmp .check_next_face_value
.return_false:
        mov eax, FALSE
        ret
.return_true:
        mov eax, TRUE
        ret

count_face:
        mov eax, 0
        cmp ebx, [esi+0]
        jne .next1
        inc eax
.next1:
        cmp ebx, [esi+4]
        jne .next2
        inc eax
.next2:
        cmp ebx, [esi+8]
        jne .next3
        inc eax
.next3:
        cmp ebx, [esi+12]
        jne .next4
        inc eax
.next4:
        cmp ebx, [esi+16]
        jne .next5
        inc eax
.next5:
        ret

sum_of_dice:
        mov eax, [esi+0]
        add eax, [esi+4]
        add eax, [esi+8]
        add eax, [esi+12]
        add eax, [esi+16]
        ret
```



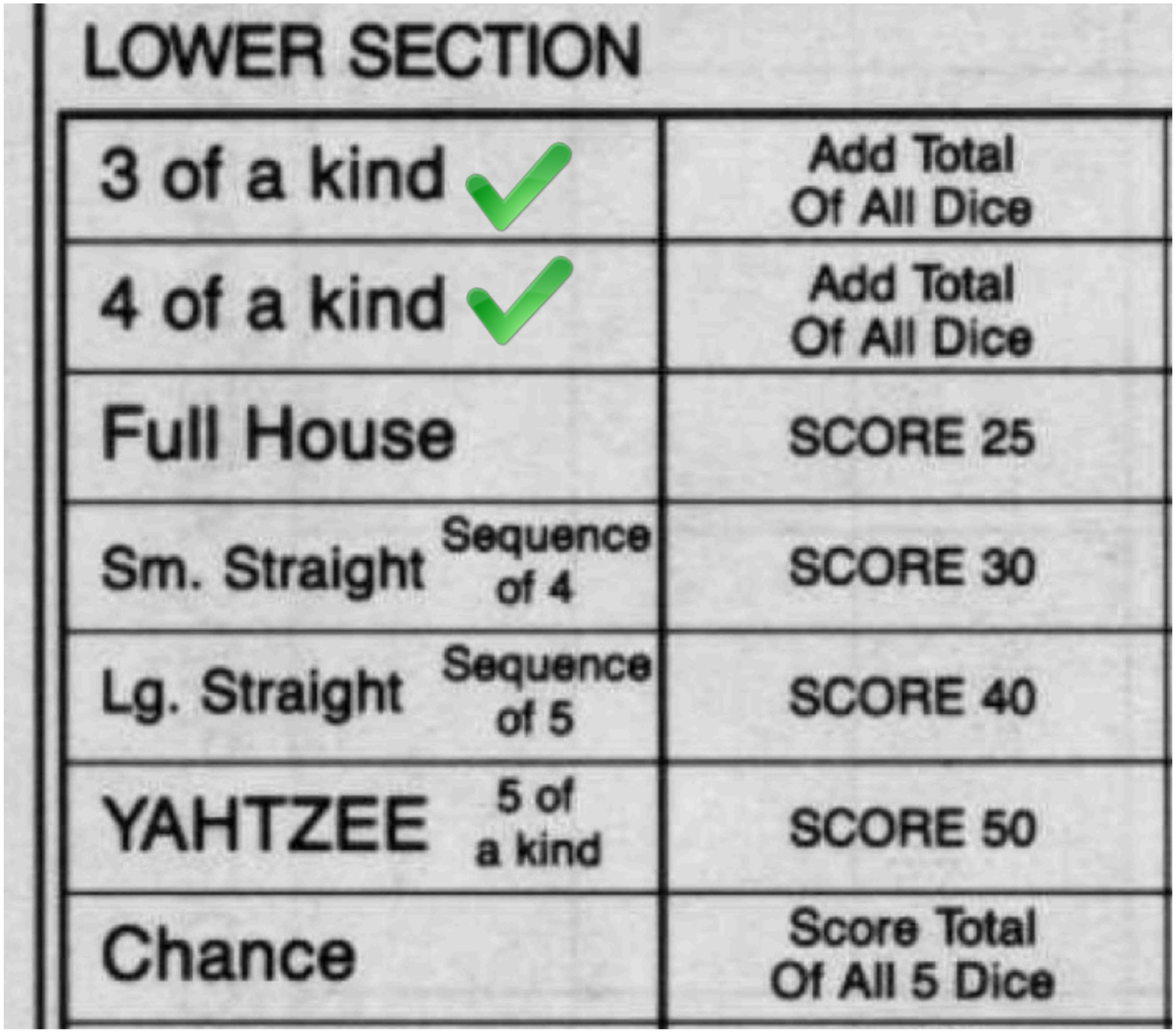
```
%define TRUE 1
%define FALSE 0

global score_three_of_a_kind
score_three_of_a_kind:
        call have_at_least_3_of_a_kind
        cmp eax, TRUE
        je .return_sum
.return_zero:
        mov eax, 0
        ret
.return_sum:
        call sum_of_dice
        ret

have_at_least_3_of_a_kind:
        mov ebx, 1 ; face value
.check_next_face_value:
        call count_face
        cmp eax, 3
        jge .return_true
        inc ebx
        cmp ebx, 6
        jg .return_false
        jmp .check_next_face_value
.return_false:
        mov eax, FALSE
        ret
.return_true:
        mov eax, TRUE
        ret

count_face:
        mov eax, 0
        cmp ebx, [esi+0]
        jne .next1
        inc eax
.next1:
        cmp ebx, [esi+4]
        jne .next2
        inc eax
.next2:
        cmp ebx, [esi+8]
        jne .next3
        inc eax
.next3:
        cmp ebx, [esi+12]
        jne .next4
        inc eax
.next4:
        cmp ebx, [esi+16]
        jne .next5
        inc eax
.next5:
        ret

sum_of_dice:
        mov eax, [esi+0]
        add eax, [esi+4]
        add eax, [esi+8]
        add eax, [esi+12]
        add eax, [esi+16]
        ret
```


so what about the implementation? Can we do much more here?
am I allowed to comment the code?
© www.ClipProject.info



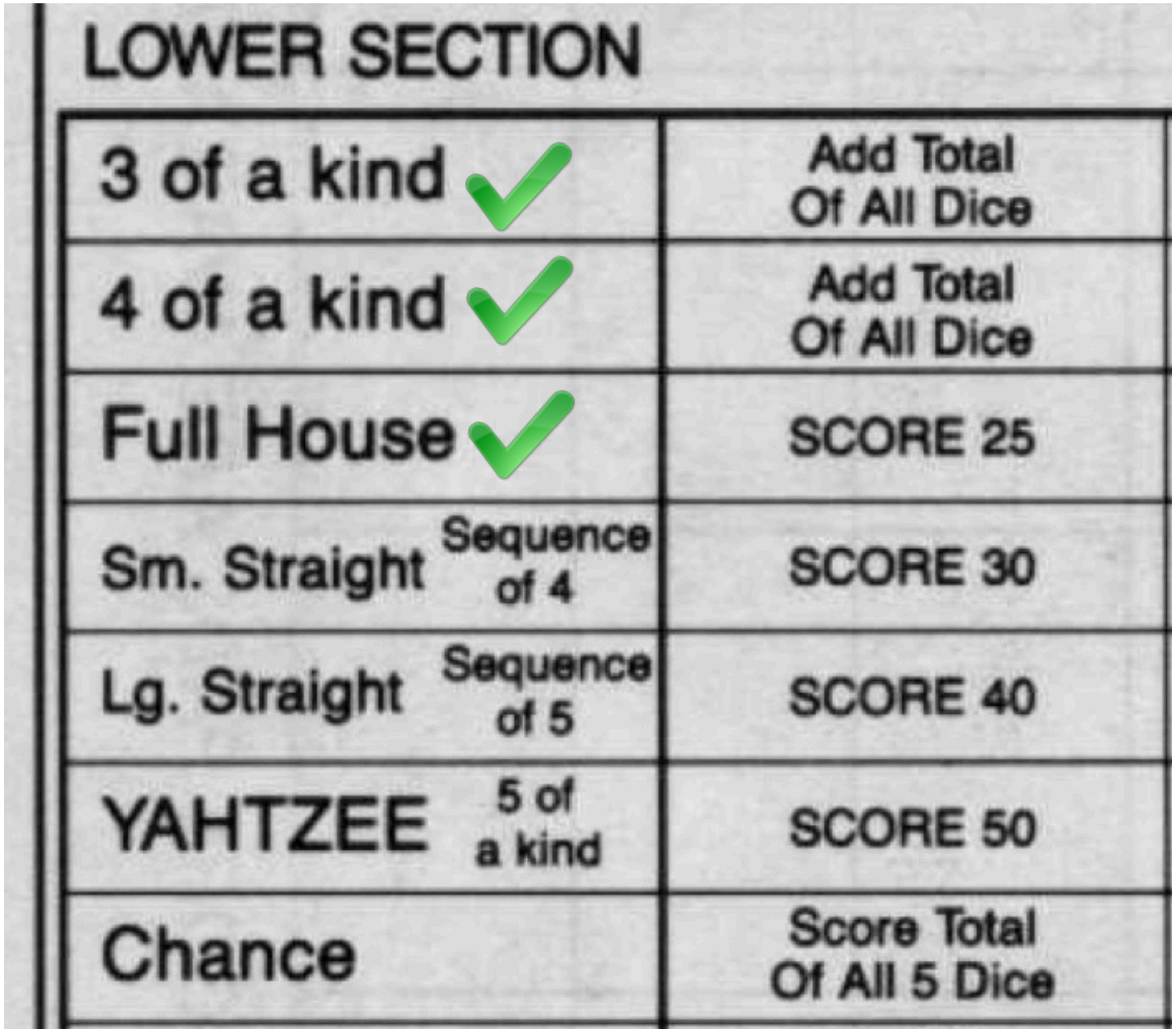
LOWER SECTION

3 of a kind ✓	Add Total Of All Dice	⚁ ⚂ ⚃ ⚃ ⚃
4 of a kind	Add Total Of All Dice	⚃ ⚄ ⚄ ⚄ ⚄
Full House	SCORE 25	⚁ ⚁ ⚄ ⚄ ⚄
Sm. Straight Sequence of 4	SCORE 30	⚀ ⚂ ⚃ ⚄ ⚅
Lg. Straight Sequence of 5	SCORE 40	⚀ ⚁ ⚂ ⚃ ⚄
YAHTZEE 5 of a kind	SCORE 50	⚂ ⚂ ⚂ ⚂ ⚂
Chance	Score Total Of All 5 Dice	⚀ ⚀ ⚂ ⚂ ⚄



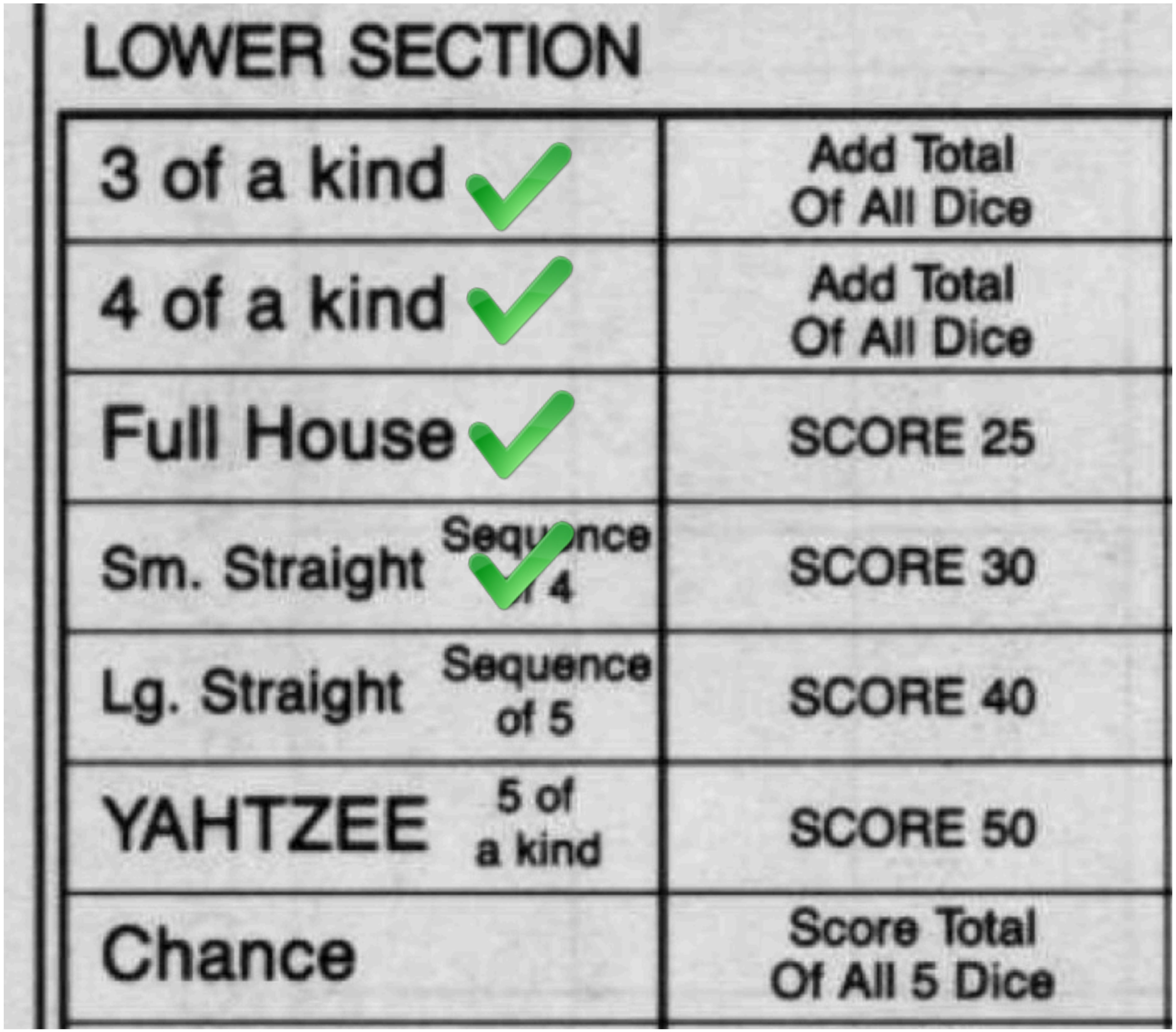
LOWER SECTION

3 of a kind ✓	Add Total Of All Dice	⚁ ⚂ ⚃ ⚃ ⚃
4 of a kind ✓	Add Total Of All Dice	⚃ ⚄ ⚄ ⚄ ⚄
Full House	SCORE 25	⚁ ⚁ ⚄ ⚄ ⚄
Sm. Straight Sequence of 4	SCORE 30	⚀ ⚂ ⚃ ⚄ ⚅
Lg. Straight Sequence of 5	SCORE 40	⚀ ⚁ ⚂ ⚃ ⚄
YAHTZEE 5 of a kind	SCORE 50	⚂ ⚂ ⚂ ⚂ ⚂
Chance	Score Total Of All 5 Dice	⚀ ⚀ ⚂ ⚂ ⚄



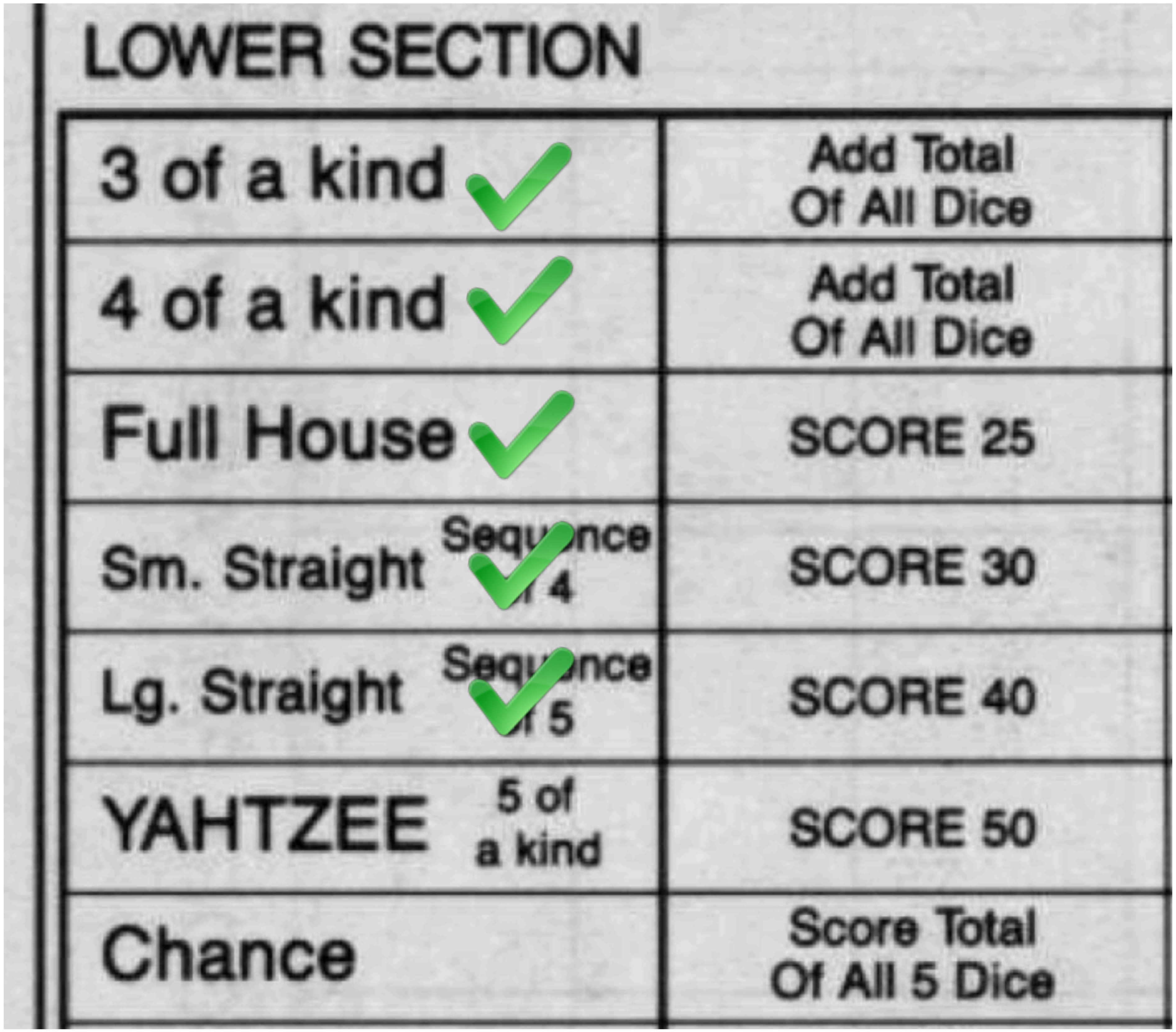
LOWER SECTION

3 of a kind ✓	Add Total Of All Dice	⚁ ⚂ ⚃ ⚃ ⚃
4 of a kind ✓	Add Total Of All Dice	⚃ ⚄ ⚄ ⚄ ⚄
Full House ✓	SCORE 25	⚁ ⚁ ⚄ ⚄ ⚄
Sm. Straight Sequence of 4	SCORE 30	⚀ ⚂ ⚃ ⚄ ⚅
Lg. Straight Sequence of 5	SCORE 40	⚀ ⚁ ⚂ ⚃ ⚄
YAHTZEE 5 of a kind	SCORE 50	⚂ ⚂ ⚂ ⚂ ⚂
Chance	Score Total Of All 5 Dice	⚀ ⚀ ⚂ ⚂ ⚄



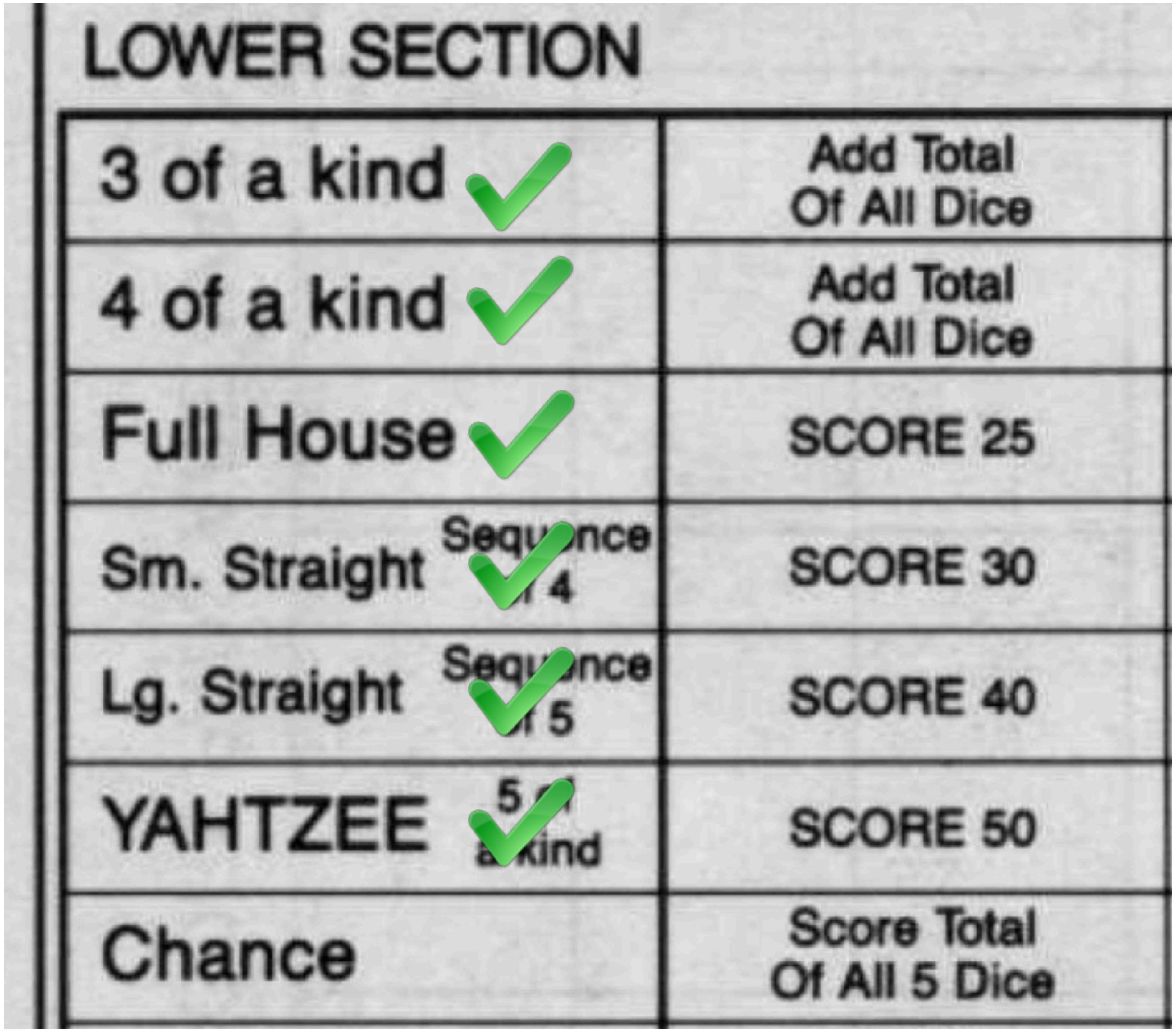
LOWER SECTION

3 of a kind	✓	Add Total Of All Dice	⚁	⚂	⚃	⚃	⚃
4 of a kind	✓	Add Total Of All Dice	⚃	⚄	⚄	⚄	⚄
Full House	✓	SCORE 25	⚁	⚁	⚄	⚄	⚄
Sm. Straight Sequence of 4	✓	SCORE 30	⚀	⚂	⚃	⚄	⚅
Lg. Straight Sequence of 5		SCORE 40	⚀	⚁	⚂	⚃	⚄
YAHTZEE 5 of a kind		SCORE 50	⚂	⚂	⚂	⚂	⚂
Chance		Score Total Of All 5 Dice	⚀	⚀	⚂	⚂	⚄



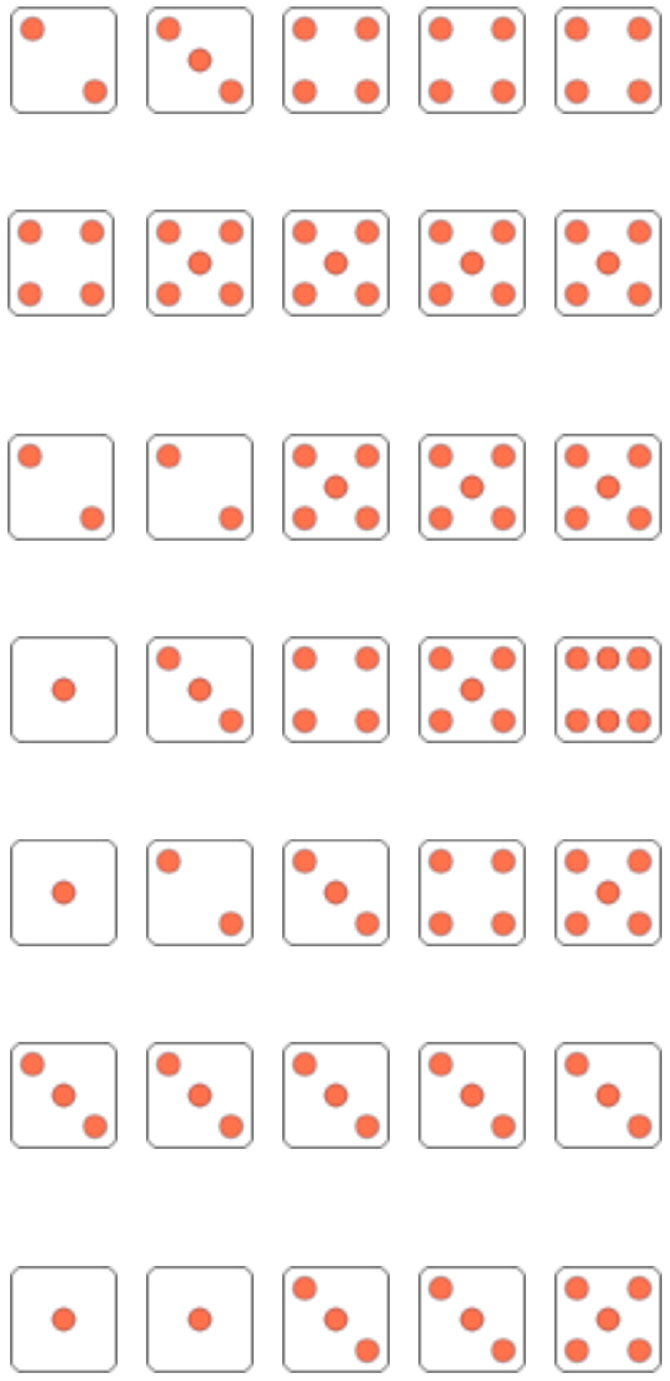
LOWER SECTION

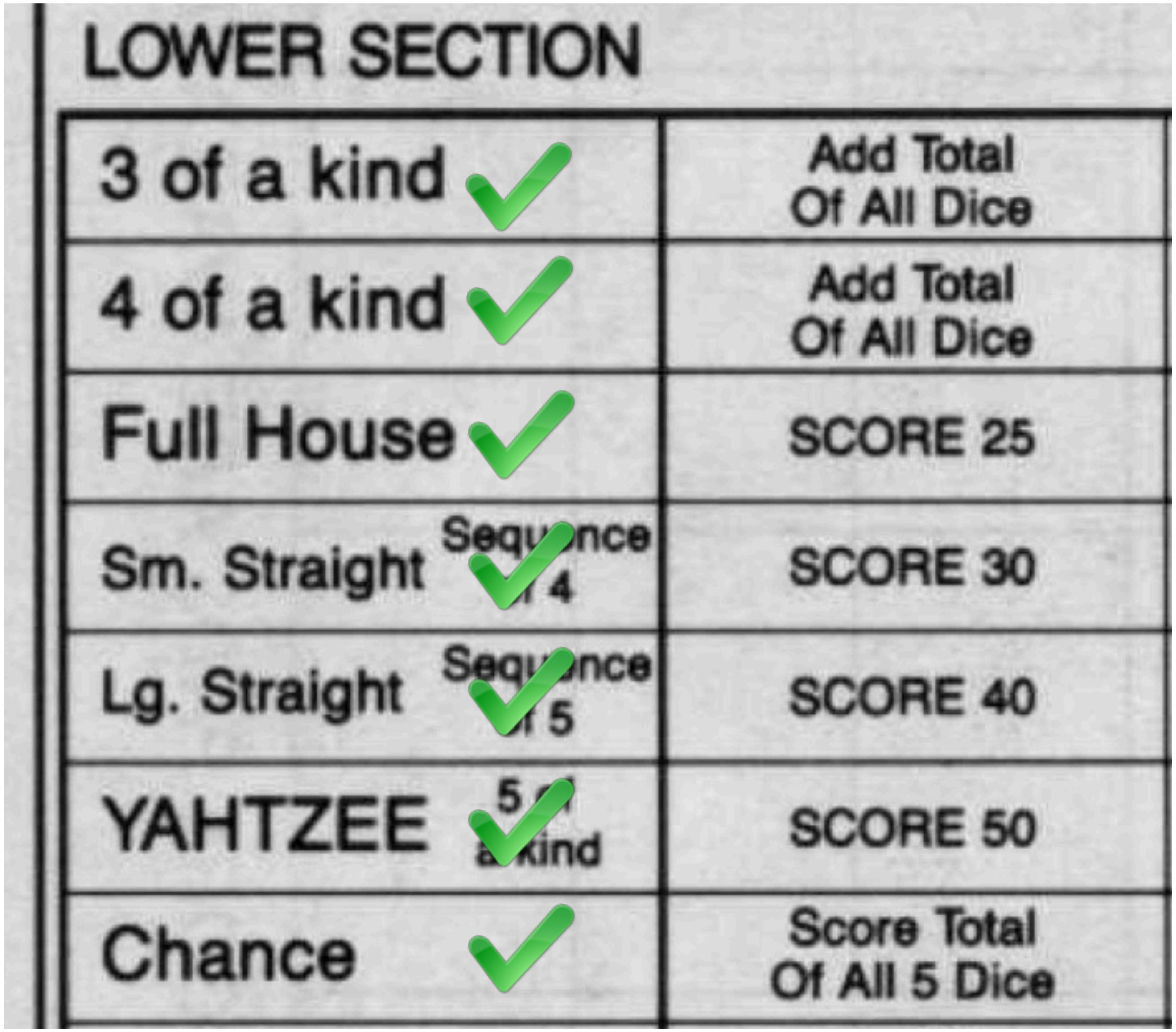
3 of a kind ✓	Add Total Of All Dice	⚁ ⚂ ⚃ ⚃ ⚃
4 of a kind ✓	Add Total Of All Dice	⚃ ⚄ ⚄ ⚄ ⚄
Full House ✓	SCORE 25	⚁ ⚁ ⚄ ⚄ ⚄
Sm. Straight Sequence of 4 ✓	SCORE 30	⚀ ⚂ ⚃ ⚄ ⚅
Lg. Straight Sequence of 5 ✓	SCORE 40	⚀ ⚁ ⚂ ⚃ ⚄
YAHTZEE 5 of a kind	SCORE 50	⚂ ⚂ ⚂ ⚂ ⚂
Chance	Score Total Of All 5 Dice	⚀ ⚀ ⚂ ⚂ ⚄



LOWER SECTION

3 of a kind ✓	Add Total Of All Dice
4 of a kind ✓	Add Total Of All Dice
Full House ✓	SCORE 25
Sm. Straight Sequence of 4 ✓	SCORE 30
Lg. Straight Sequence of 5 ✓	SCORE 40
YAHTZEE 5 of a kind ✓	SCORE 50
Chance	Score Total Of All 5 Dice





LOWER SECTION

3 of a kind ✓	Add Total Of All Dice	⚁ ⚂ ⚃ ⚃ ⚃
4 of a kind ✓	Add Total Of All Dice	⚃ ⚄ ⚄ ⚄ ⚄
Full House ✓	SCORE 25	⚁ ⚁ ⚄ ⚄ ⚄
Sm. Straight Sequence of 4 ✓	SCORE 30	⚀ ⚂ ⚃ ⚄ ⚅
Lg. Straight Sequence of 5 ✓	SCORE 40	⚀ ⚁ ⚂ ⚃ ⚄
YAHTZEE 5 of a kind ✓	SCORE 50	⚂ ⚂ ⚂ ⚂ ⚂
Chance ✓	Score Total Of All 5 Dice	⚀ ⚀ ⚂ ⚂ ⚄

Larrys ad-hoc unit test framework for assembler (aka, OSL)

```
;
; Larrys adhoc unit test framework (aka, OSL)
;

section .data
        TEST_tests dd 0
        TEST_errflag dd 0
        TEST_errors dd 0
        TEST_okchar db '.'
        TEST_errchar db 'X'
        TEST_okstr db ' OK', 0xa
        TEST_okstr_len equ $-TEST_okstr
        TEST_errstr db ' FAILED', 0xa
        TEST_errstr_len equ $-TEST_errstr
section .text

%macro TEST_runtests 1
section .data
        %defstr %1_str %1
        %%prefix db %1_str, ' '
        %%prefix_len equ $-%%prefix
section .text
        sys_write STDOUT, %%prefix, %%prefix_len
        mov [TEST_errflag], dword 0
        call %1
        cmp [TEST_errflag], dword 0
        jne %%print_err
        sys_write STDOUT, TEST_okstr, TEST_okstr_len
        jmp %%cont
%%print_err:
        sys_write STDOUT, TEST_errstr, TEST_errstr_len
%%cont:
%endmacro
```

```
%macro TEST_assert_eax_equals 1
        sub eax, dword %1
        jz %%ok
        add [TEST_tests], dword 1
        add [TEST_errors], dword 1
        add [TEST_errflag], dword 1
        sys_write STDOUT, TEST_errchar, 1
        jmp %%exit
%%ok:
        add [TEST_tests], dword 1
        sys_write STDOUT, TEST_okchar, 1
%%exit:
        nop
%endmacro

%macro TEST_exit 0
        cmp [TEST_errors], dword 0
        jne .exit_success
.exit_success:
        sys_exit EXIT_SUCCESS
.exit_failure:
        sys_exit EXIT_FAILURE
%endmacro
```

Larrys ad-hoc unit test framework for assembler (aka, OSL)

```
%macro TEST_print_eax 0
section .data
        %%ch db '.'
section .text
        push eax
        push ebx
        push ecx
        push edx

        mov ebx, 0
%%divide_and_push_remainder:
        mov edx, 0
        mov ecx, 10
        div ecx ; eax = quotient, edx = remainder
        add edx, '0'
        push edx
        inc ebx
        cmp eax, 0
        jne %%divide_and_push_remainder
%%pop_and_print:
        pop eax
        mov [%%ch], al
        sys_write STDOUT, %%ch, 1
        dec ebx
        jg %%pop_and_print
%%exit:
        pop edx
        pop ecx
        pop ebx
        pop eax
%endmacro
```

```
%macro TEST_print_summary 0
section .data
        %%str0 db 'CHECK ', __FILE__, " -> "
        %%str0_len equ $-%%str0
        %%str1 db " tests executed. "
        %%str1_len equ $-%%str1
        %%str2 db " failed.)", 0xa
        %%str2_len equ $-%%str2
        %%str3 db "OK ("
        %%str3_len equ $-%%str3
        %%str4 db "FAILED ("
        %%str4_len equ $-%%str4
section .text

        sys_write STDOUT, %%str0, %%str0_len

        cmp [TEST_errors], dword 0
        jne .print_failure
        sys_write STDOUT, %%str3, %%str3_len
        jmp .cont
.print_failure:
        sys_write STDOUT, %%str4, %%str4_len
.cont:
        push eax
        mov eax, [TEST_tests]
        TEST_print_eax
        sys_write STDOUT, %%str1, %%str1_len

        mov eax, [TEST_errors]
        TEST_print_eax
        sys_write STDOUT, %%str2, %%str2_len
        pop eax
%endmacro
```

the unit tests for the yahtzee library

```
%include "mylib.inc"
%include "mytest.inc"

%macro eval_dice 6
section .data
        %%dice dd %2,%3,%4,%5,%6
section .text
        mov esi, %%dice
        call %1
%endmacro

;
; functions to test
;

extern score_chance
extern score_yahtzee
extern score_three_of_a_kind
extern score_four_of_a_kind
extern score_small_straight
extern score_large_straight
extern score_full_house

;
; test functions
;

check_score_chance:
        eval_dice score_chance, 1,1,1,1,1
        TEST_assert_eax_equals 5

        eval_dice score_chance, 1,1,1,1,1
        TEST_assert_eax_equals 5

        eval_dice score_chance, 1,1,1,1,2
        TEST_assert_eax_equals 6

        eval_dice score_chance, 6,6,6,6,6
        TEST_assert_eax_equals 30

        eval_dice score_chance, 1,2,3,4,5
        TEST_assert_eax_equals 15

        eval_dice score_chance, 6,5,4,3,2
        TEST_assert_eax_equals 20

        ret

check_score_yahtzee:
        eval_dice score_yahtzee, 1,1,1,1,1
        TEST_assert_eax_equals 50

        eval_dice score_yahtzee, 1,2,3,4,5
        TEST_assert_eax_equals 0

        eval_dice score_yahtzee, 1,1,1,1,2
        TEST_assert_eax_equals 0

        eval_dice score_yahtzee, 6,6,6,6,6
        TEST_assert_eax_equals 50

        ret

check_score_three_of_a_kind:
        eval_dice score_three_of_a_kind, 1,1,1,1,1
        TEST_assert_eax_equals 5

        eval_dice score_three_of_a_kind, 1,1,1,1,2
        TEST_assert_eax_equals 6

        eval_dice score_three_of_a_kind, 1,2,3,4,5
        TEST_assert_eax_equals 0

        eval_dice score_three_of_a_kind, 6,6,6,6,6
        TEST_assert_eax_equals 30

        eval_dice score_three_of_a_kind, 6,5,4,3,2
        TEST_assert_eax_equals 0

        ret
```

```
check_score_four_of_a_kind:
        eval_dice score_four_of_a_kind, 1,1,1,1,1
        TEST_assert_eax_equals 5

        eval_dice score_four_of_a_kind, 1,1,1,1,2
        TEST_assert_eax_equals 6

        eval_dice score_four_of_a_kind, 6,6,6,6,6
        TEST_assert_eax_equals 30

        eval_dice score_four_of_a_kind, 1,2,3,4,5
        TEST_assert_eax_equals 0

        ret

check_score_small_straight:
        eval_dice score_small_straight, 1,2,3,4,5
        TEST_assert_eax_equals 30

        eval_dice score_small_straight, 1,2,3,4,1
        TEST_assert_eax_equals 30

        eval_dice score_small_straight, 1,2,4,5,6
        TEST_assert_eax_equals 0

        eval_dice score_small_straight, 6,5,4,3,1
        TEST_assert_eax_equals 30

        eval_dice score_small_straight, 6,6,6,6,6
        TEST_assert_eax_equals 0

        ret

check_score_large_straight:
        eval_dice score_large_straight, 1,2,3,4,5
        TEST_assert_eax_equals 40

        eval_dice score_large_straight, 6,6,6,6,6
        TEST_assert_eax_equals 0

        eval_dice score_large_straight, 6,5,4,3,2
        TEST_assert_eax_equals 40

        ret

check_score_full_house:
        eval_dice score_full_house, 3,3,3,5,5
        TEST_assert_eax_equals 25

        eval_dice score_full_house, 3,5,3,5,5
        TEST_assert_eax_equals 25

        eval_dice score_full_house, 6,6,6,6,6
        TEST_assert_eax_equals 0

        eval_dice score_full_house, 2,3,2,3,1
        TEST_assert_eax_equals 0

        ret

global start
start:
        TEST_runtests check_score_chance
        TEST_runtests check_score_yahtzee
        TEST_runtests check_score_three_of_a_kind
        TEST_runtests check_score_four_of_a_kind
        TEST_runtests check_score_small_straight
        TEST_runtests check_score_large_straight
        TEST_runtests check_score_full_house

        TEST_print_summary
        TEST_exit
```

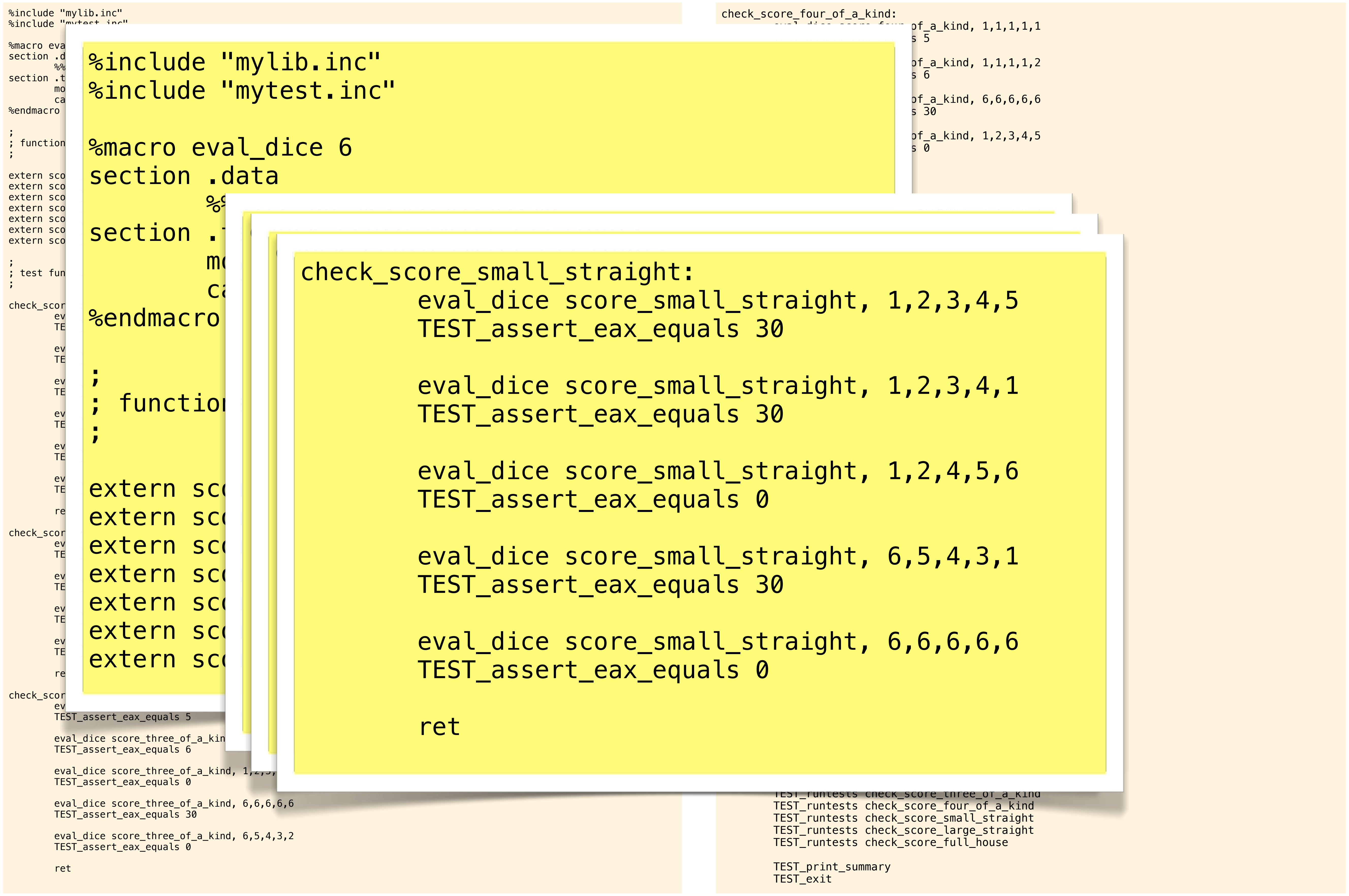
the unit tests for the yahtzee library

```
%include "mylib.inc"
%include "mytest.inc"

%macro eval_dice 6
section .data
        %%dice dd %2,%3,%4,%5,%6
section .text
        mov esi, %%dice
        call %1
%endmacro

;
; functions to test
;

extern score_chance
extern score_yahtzee
extern score_three_of_a_kind
extern score_four_of_a_kind
extern score_small_straight
extern score_large_straight
extern score_full_house
```

```
%include "mylib.inc"
%include "mytest.inc"

%macro eva
section .d
        %%
section .t
        mo
        ca
%endmacro

;
; function
;

extern sco
extern sco
extern sco
extern sco
extern sco
extern sco
extern sco

;
; test fun
;

check_scor
        ev
        TE

        ev
        TE

        ev
        TE

        ev
        TE

        ev
        TE

        ev
        TE

        re

check_scor
        ev
        TE

        ev
        TE

        ev
        TE

        ev
        TE

        re

check_scor
        ev
        TEST_assert_eax_equals 5

        eval_dice score_three_of_a_kind, 1,1,1,1,2
        TEST_assert_eax_equals 6

        eval_dice score_three_of_a_kind, 1,2,3,4,5
        TEST_assert_eax_equals 0

        eval_dice score_three_of_a_kind, 6,6,6,6,6
        TEST_assert_eax_equals 30

        eval_dice score_three_of_a_kind, 6,5,4,3,2
        TEST_assert_eax_equals 0

        ret
```

```
check_score_four_of_a_kind:
of_a_kind, 1,1,1,1,1
s 5

of_a_kind, 1,1,1,1,2
s 6

of_a_kind, 6,6,6,6,6
s 30

of_a_kind, 1,2,3,4,5
s 0

_straight, 1,2,3,4,5
s 30

_straight, 1,2,3,4,1
s 30

_straight, 1,2,4,5,6
s 0

_straight, 6,5,4,3,1
s 30

_straight, 6,6,6,6,6
s 0

_straight, 1,2,3,4,5
s 40

_straight, 6,6,6,6,6
s 0

_straight, 6,5,4,3,2
s 40

house, 3,3,3,5,5
s 25

house, 3,5,3,5,5
s 25

house, 6,6,6,6,6
s 0

house, 2,3,2,3,1
s 0
        ret

global start
start:
        TEST_runtests check_score_chance
        TEST_runtests check_score_yahtzee
        TEST_runtests check_score_three_of_a_kind
        TEST_runtests check_score_four_of_a_kind
        TEST_runtests check_score_small_straight
        TEST_runtests check_score_large_straight
        TEST_runtests check_score_full_house

        TEST_print_summary
        TEST_exit
```

the unit tests for the yahtzee library



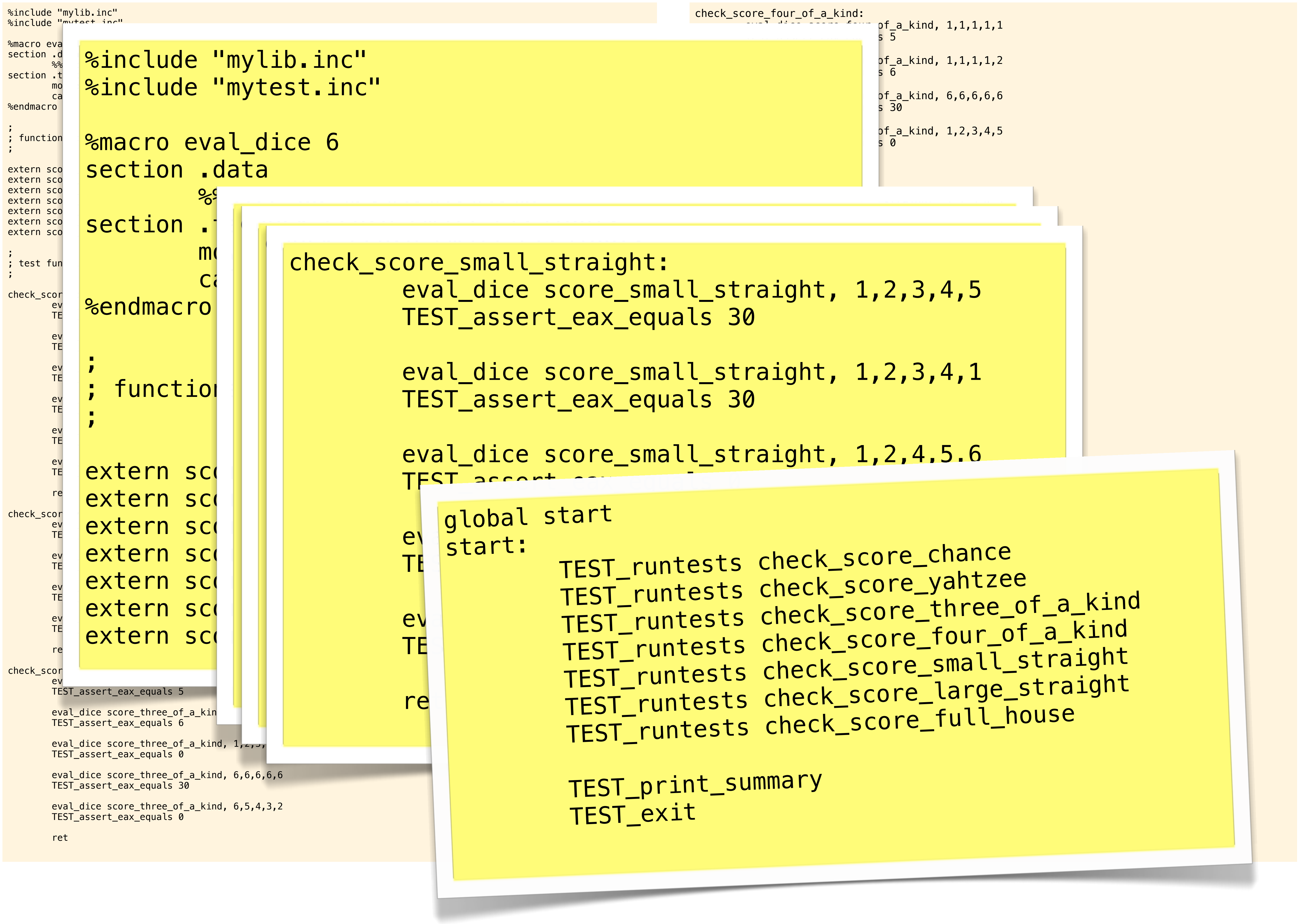
```
%include "mylib.inc"
%include "mytest.inc"

%macro eval_dice 6
section .data
        %%
section .t
        mo
        ca
%endmacro

;
; function
;

extern sco
extern sco
extern sco
extern sco
extern sco
extern sco
extern sco
```

```
check_score_small_straight:
        eval_dice score_small_straight, 1,2,3,4,5
        TEST_assert_eax_equals 30

        eval_dice score_small_straight, 1,2,3,4,1
        TEST_assert_eax_equals 30

        eval_dice score_small_straight, 1,2,4,5,6
        TEST_assert_eax_equals 0

        eval_dice score_small_straight, 6,5,4,3,1
        TEST_assert_eax_equals 30

        eval_dice score_small_straight, 6,6,6,6,6
        TEST_assert_eax_equals 0

        ret
```

```
check_scor
        ev
        TEST_assert_eax_equals 5

        eval_dice score_three_of_a_kin
        TEST_assert_eax_equals 6

        eval_dice score_three_of_a_kind, 1,2,3,
        TEST_assert_eax_equals 0

        eval_dice score_three_of_a_kind, 6,6,6,6,6
        TEST_assert_eax_equals 30

        eval_dice score_three_of_a_kind, 6,5,4,3,2
        TEST_assert_eax_equals 0

        ret
```

```
check_score_four_of_a_kind:
        of_a_kind, 1,1,1,1,1
        s 5

        of_a_kind, 1,1,1,1,2
        s 6

        of_a_kind, 6,6,6,6,6
        s 30

        of_a_kind, 1,2,3,4,5
        s 0

        TEST_runtests check_score_four_of_a_kind
        TEST_runtests check_score_small_straight
        TEST_runtests check_score_large_straight
        TEST_runtests check_score_full_house

        TEST_print_summary
        TEST_exit
```

the unit tests for the yahtzee library



```
%include "mylib.inc"
%include "mytest.inc"

%macro eval_dice 6
section .data
        %%
section .t
        mo
        ca
%endmacro

;
; function
;

extern sco
extern sco
extern sco
extern sco
extern sco
extern sco
extern sco
```

```
; test fun
;

check_scor
        ev
        TE

        ev
        TE

        ev
        TE

        ev
        TE

        ev
        TE

        ev
        TE

        re
check_scor
        ev
        TE

        ev
        TE

        ev
        TE

        ev
        TE

        re
check_scor
        ev
        TEST_assert_eax_equals 5

        eval_dice score_three_of_a_kin
        TEST_assert_eax_equals 6

        eval_dice score_three_of_a_kind, 1,2,3,
        TEST_assert_eax_equals 0

        eval_dice score_three_of_a_kind, 6,6,6,6,6
        TEST_assert_eax_equals 30

        eval_dice score_three_of_a_kind, 6,5,4,3,2
        TEST_assert_eax_equals 0

        ret
```

```
check_score_four_of_a_kind:
                  of_a_kind, 1,1,1,1,1
                  s 5

                  of_a_kind, 1,1,1,1,2
                  s 6

                  of_a_kind, 6,6,6,6,6
                  s 30

                  of_a_kind, 1,2,3,4,5
                  s 0
```

```
check_score_small_straight:
        eval_dice score_small_straight, 1,2,3,4,5
        TEST_assert_eax_equals 30

        eval_dice score_small_straight, 1,2,3,4,1
        TEST_assert_eax_equals 30

        eval_dice score_small_straight, 1,2,4,5,6
        TEST_assert_eax_equals

        ev
        TE

        ev
        TE

        re
```

```
global start
start:
        TEST_runtests check_score_chance
        TEST_runtests check_score_yahtzee
        TEST_runtests check_score_three_of_a_kind
        TEST_runtests check_score_four_of_a_kind
        TEST_runtests check_score_small_straight
        TEST_runtests check_score_large_straight
        TEST_runtests check_score_full_house

        TEST_print_summary
        TEST_exit
```

Wednesday

Wednesday

are you done?

Wednesday

are you done?

I have something

Wednesday

are you done?

I have something

can you show me

Wednesday

are you done?

I have something

can you show me

eh...

```
$ for ((i=0; i<10; i++)); do make check; done
Larry is invincible!
Larry is invincible!
Larry is invincible!
Larry is invincible!
Larry is invincible!
Larry is invincible!
Larry is invincible!
Larry is invincible!
Larry is invincible!
Larry is invincible!
$
```

eh... you are finished by friday?

eh... you are finished by friday?

but....

eh... you are finished by friday?

but....

your program must be able to read a file with lots of dice, and then write all possible scores into a new file

eh... you are finished by friday?

but....

your program must be able to read a file with lots of dice, and then write all possible scores into a new file

What is the input format? What should the output format be? Any performance requirements?

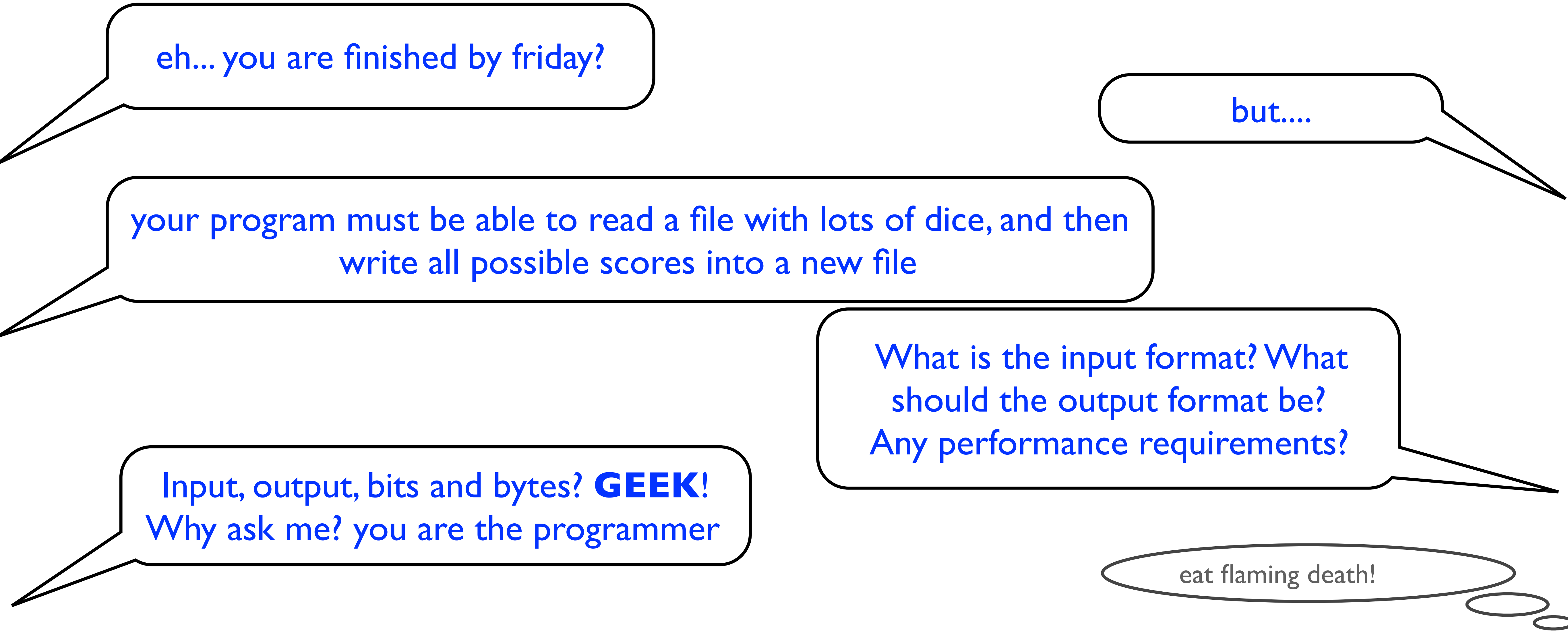
eh... you are finished by friday?

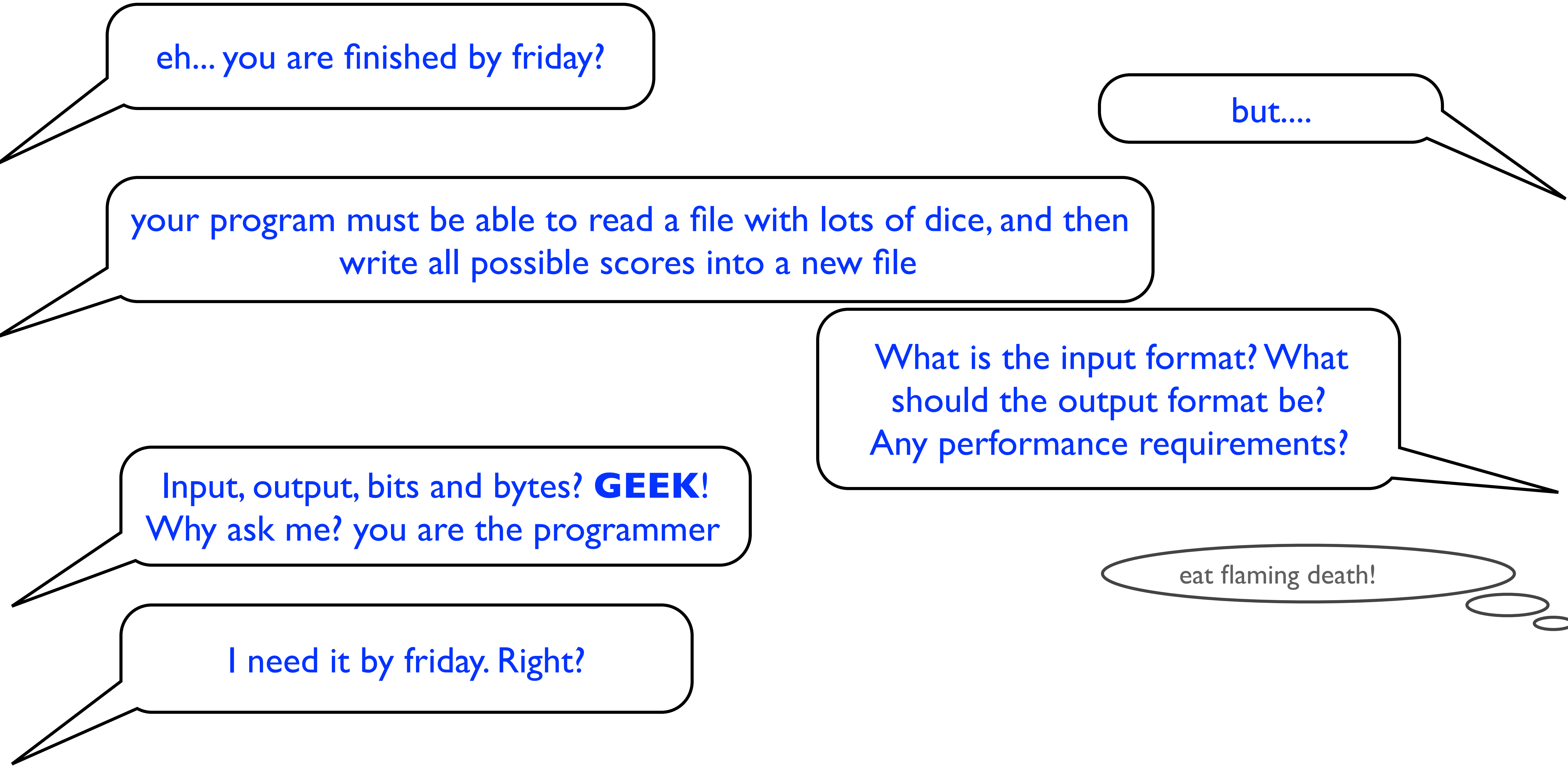
but....

your program must be able to read a file with lots of dice, and then write all possible scores into a new file

What is the input format? What should the output format be? Any performance requirements?

Input, output, bits and bytes? **GEEK**! Why ask me? you are the programmer


eh... you are finished by friday?
but....
your program must be able to read a file with lots of dice, and then write all possible scores into a new file
What is the input format? What should the output format be? Any performance requirements?
Input, output, bits and bytes? **GEEK**! Why ask me? you are the programmer
eat flaming death!


eh... you are finished by friday?
but....
your program must be able to read a file with lots of dice, and then write all possible scores into a new file
What is the input format? What should the output format be? Any performance requirements?
Input, output, bits and bytes? **GEEK**! Why ask me? you are the programmer
eat flaming death!
I need it by friday. Right?

eh... you are finished by friday?

but....

your program must be able to read a file with lots of dice, and then write all possible scores into a new file

What is the input format? What should the output format be? Any performance requirements?

Input, output, bits and bytes? **GEEK**! Why ask me? you are the programmer

eat flaming death!

I need it by friday. Right?

sure

.... Larry implements the yahtzee_demo ...
(using FDD)

Thursday

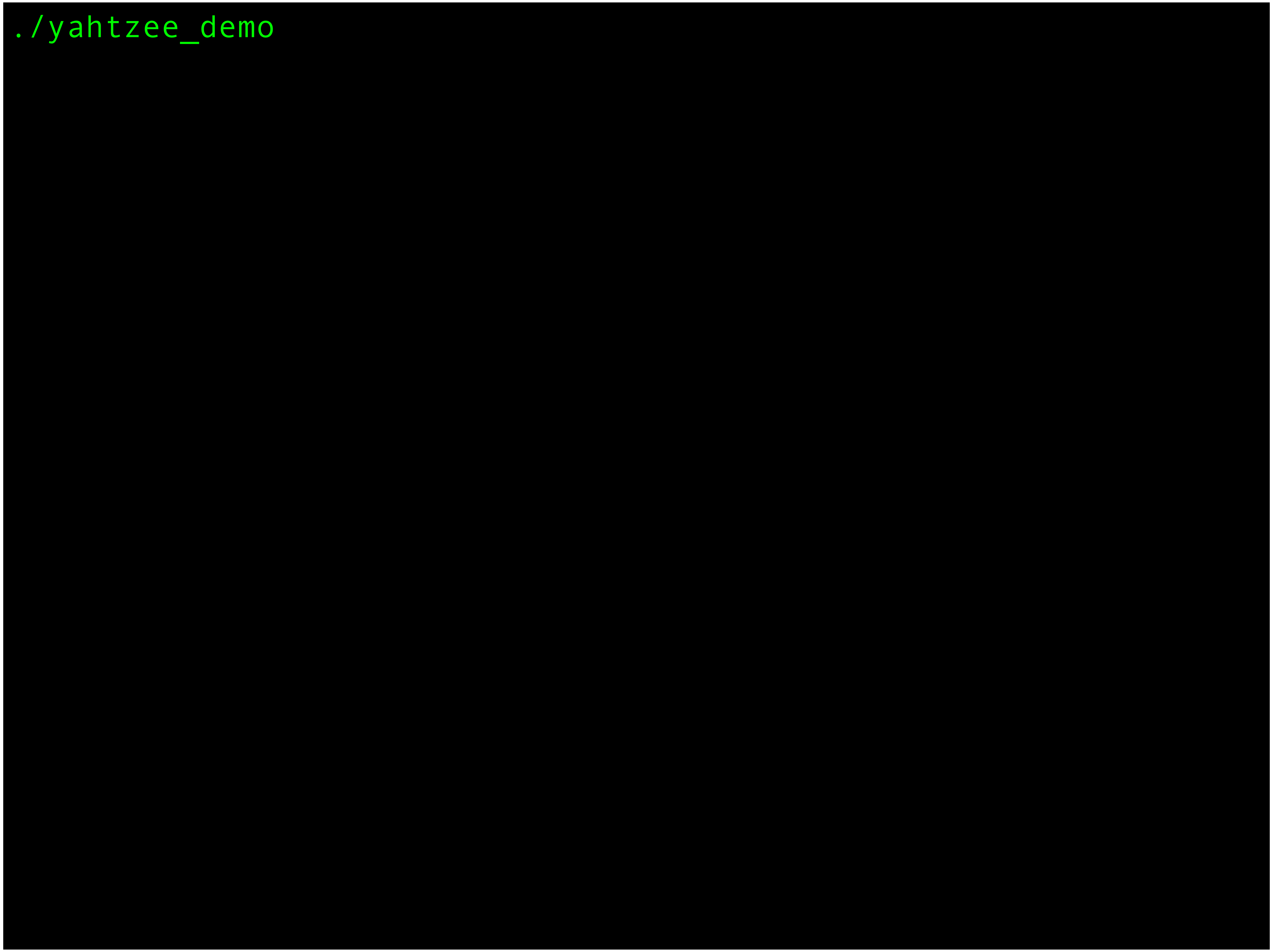
Thursday

can you show me something

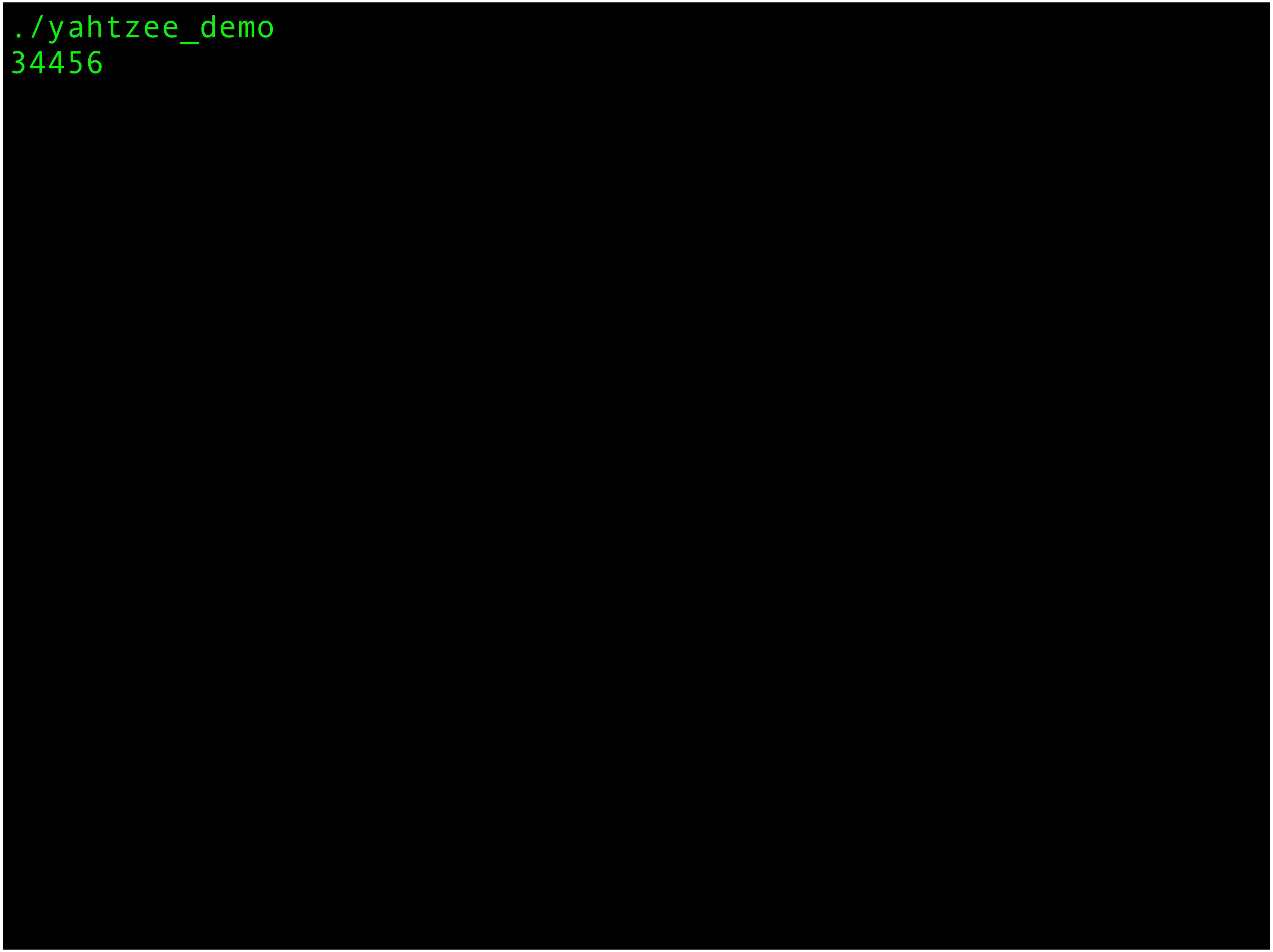
Thursday

can you show me something

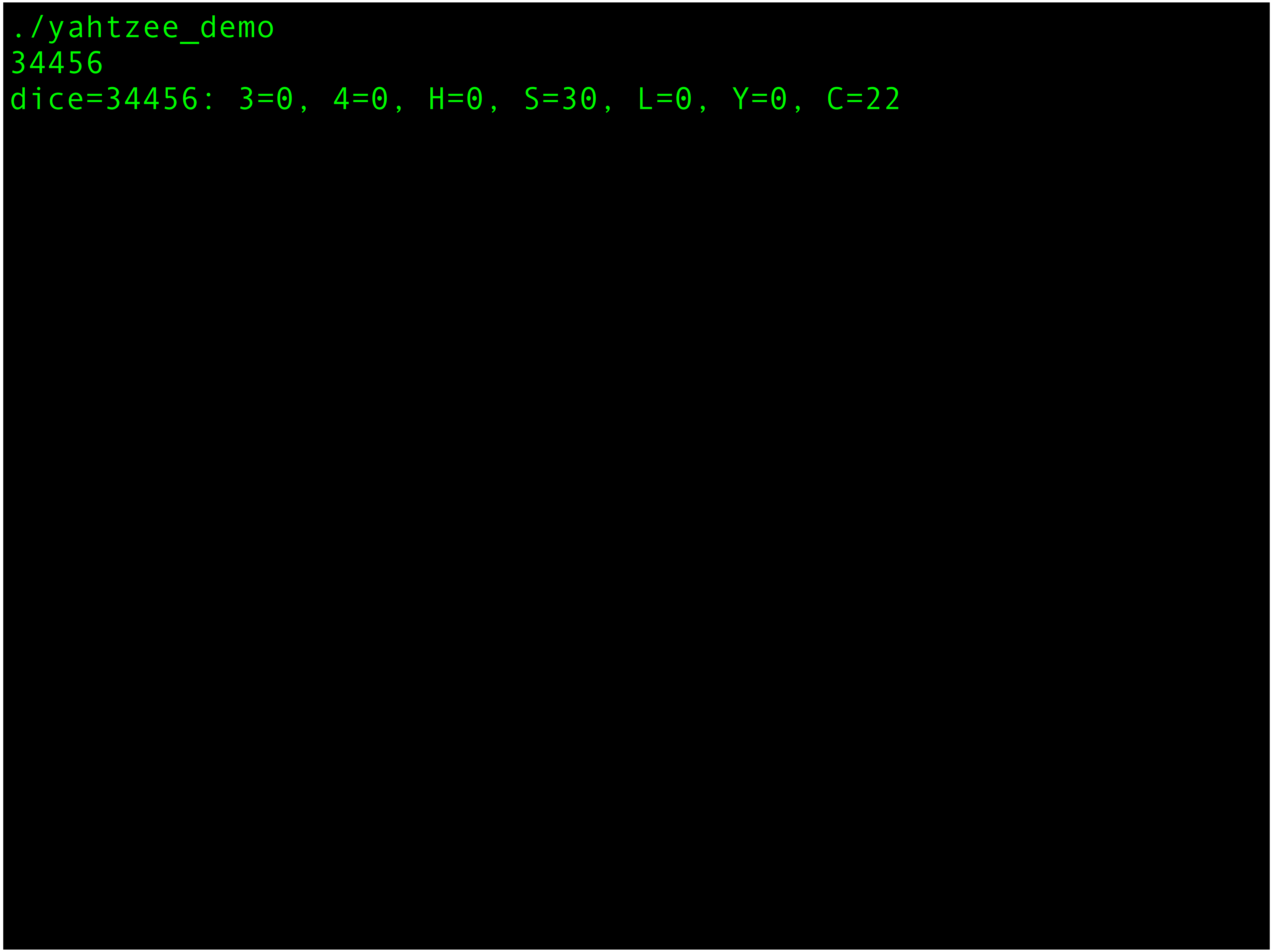
yes, here is a demo



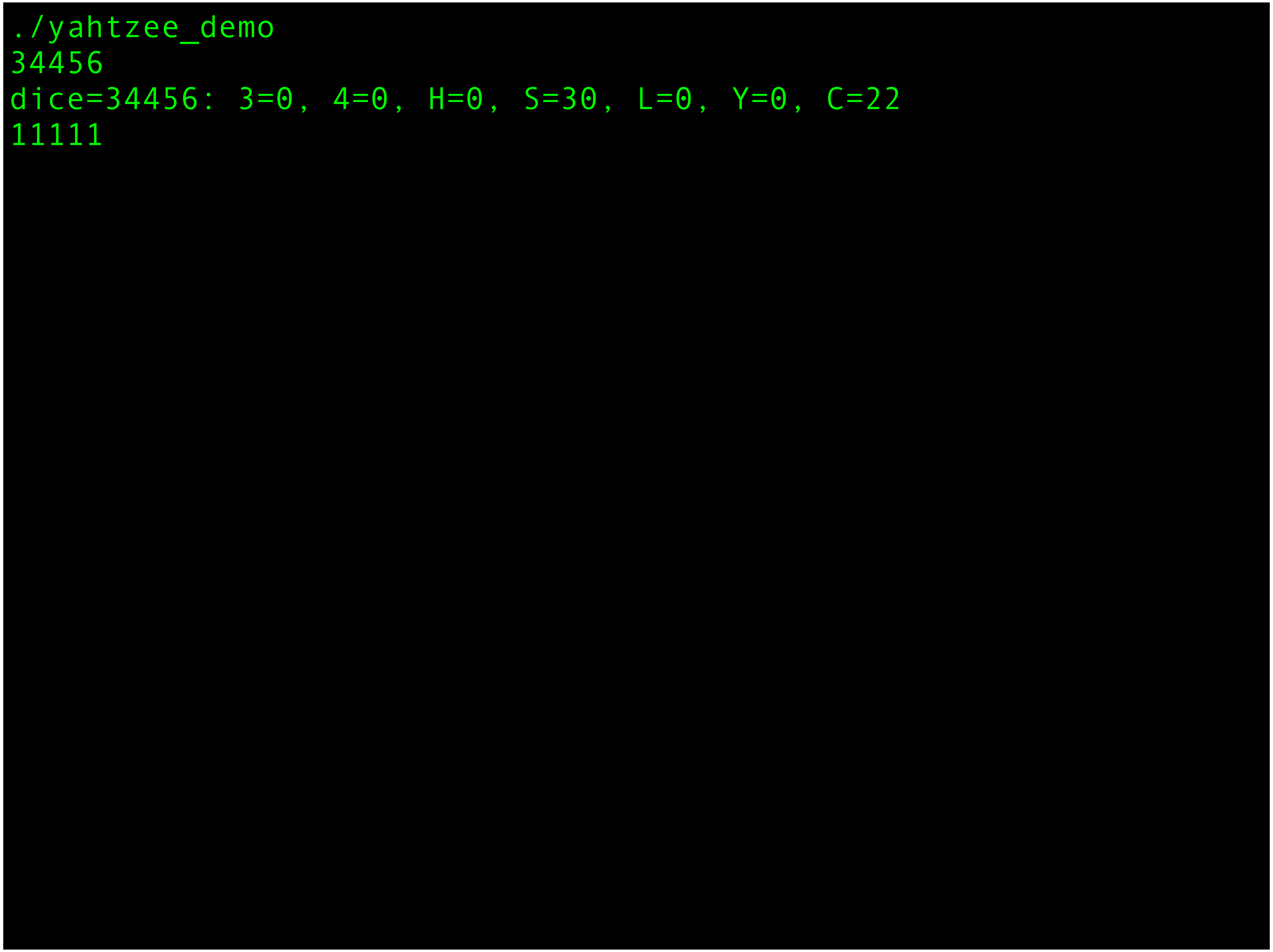
```
./yahtzee_demo
```



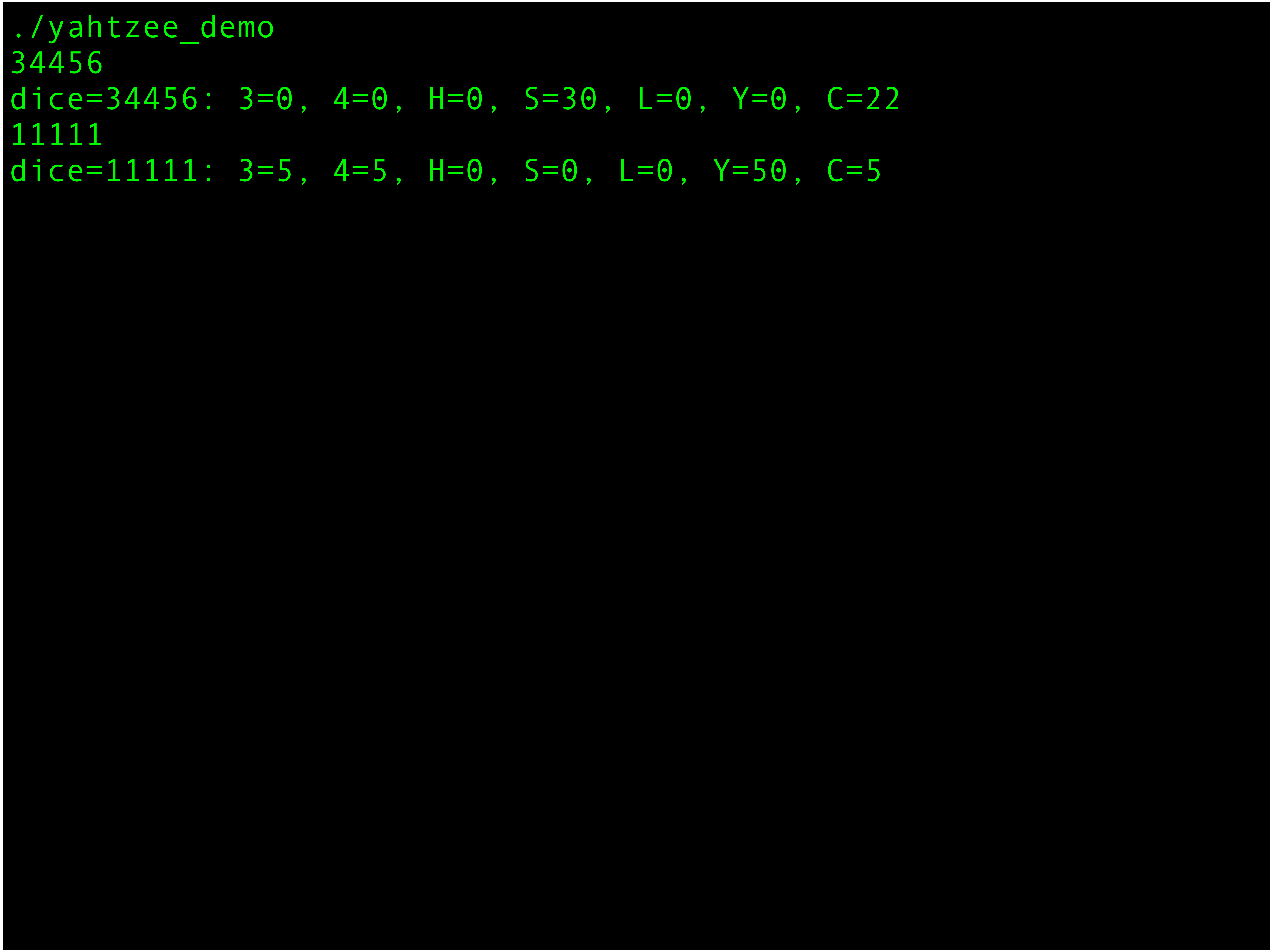
```
./yahtzee_demo
34456
```



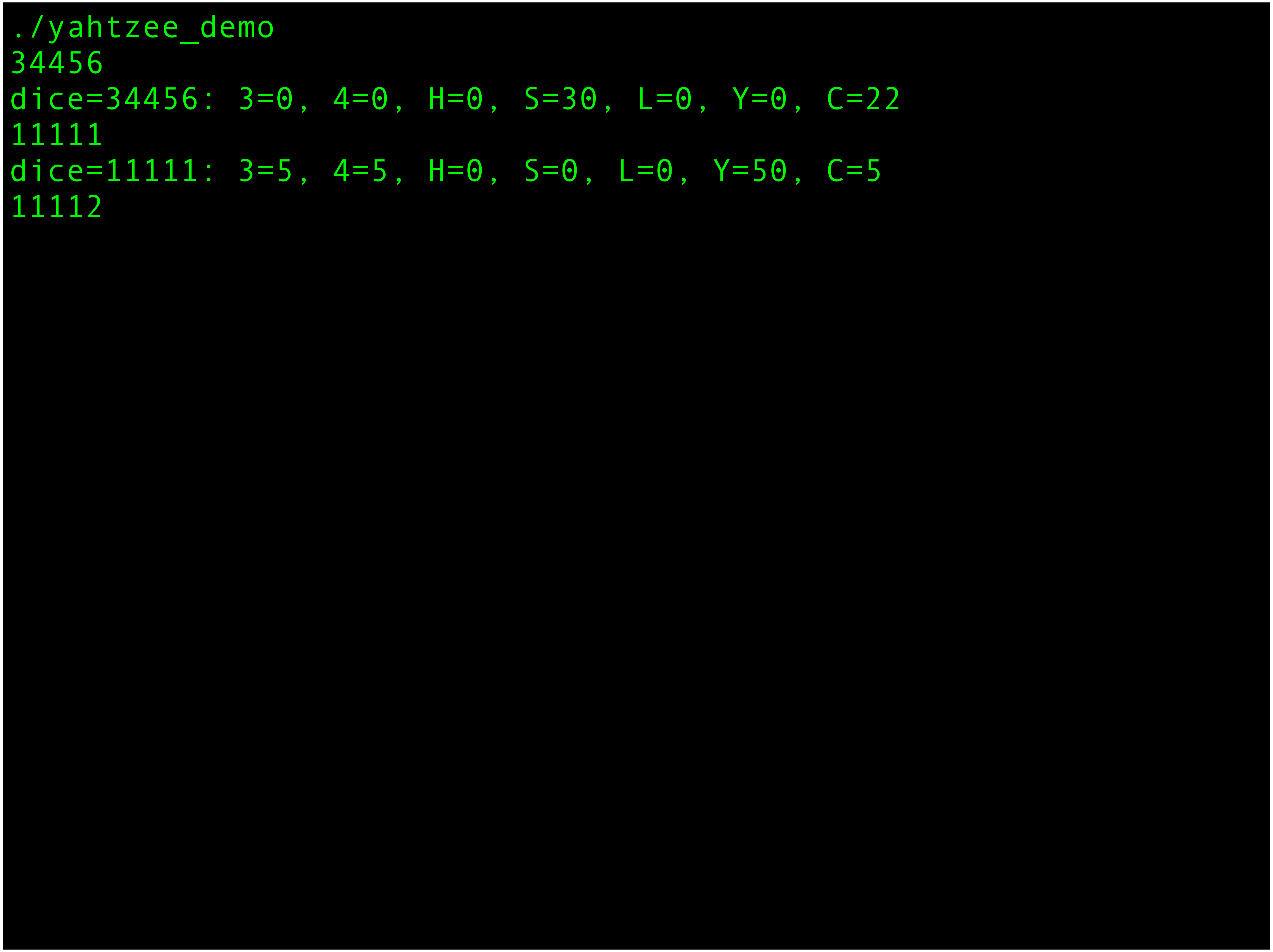
```
./yahtzee_demo
34456
dice=34456: 3=0, 4=0, H=0, S=30, L=0, Y=0, C=22
```



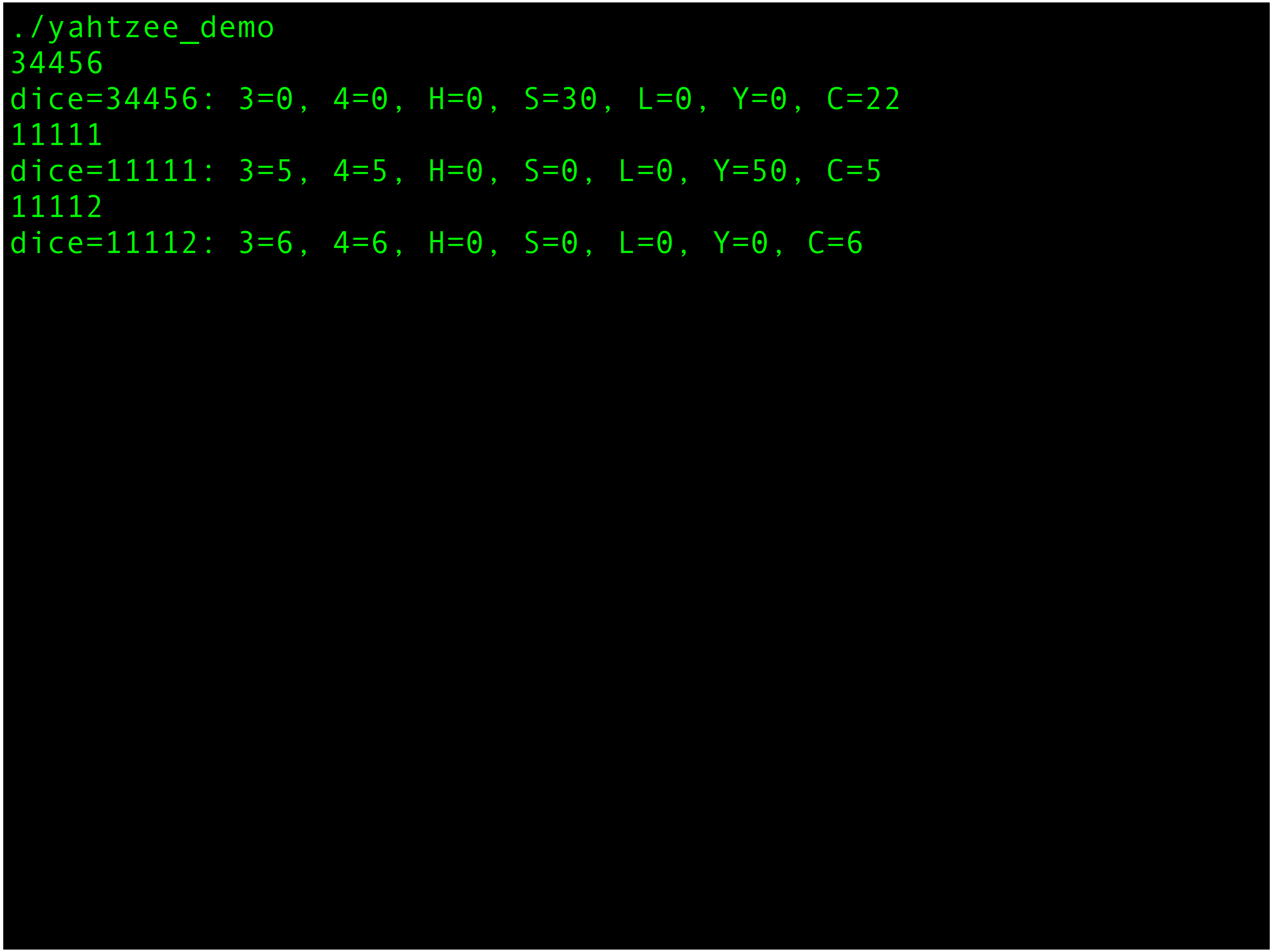
```
./yahtzee_demo
34456
dice=34456: 3=0, 4=0, H=0, S=30, L=0, Y=0, C=22
11111
```



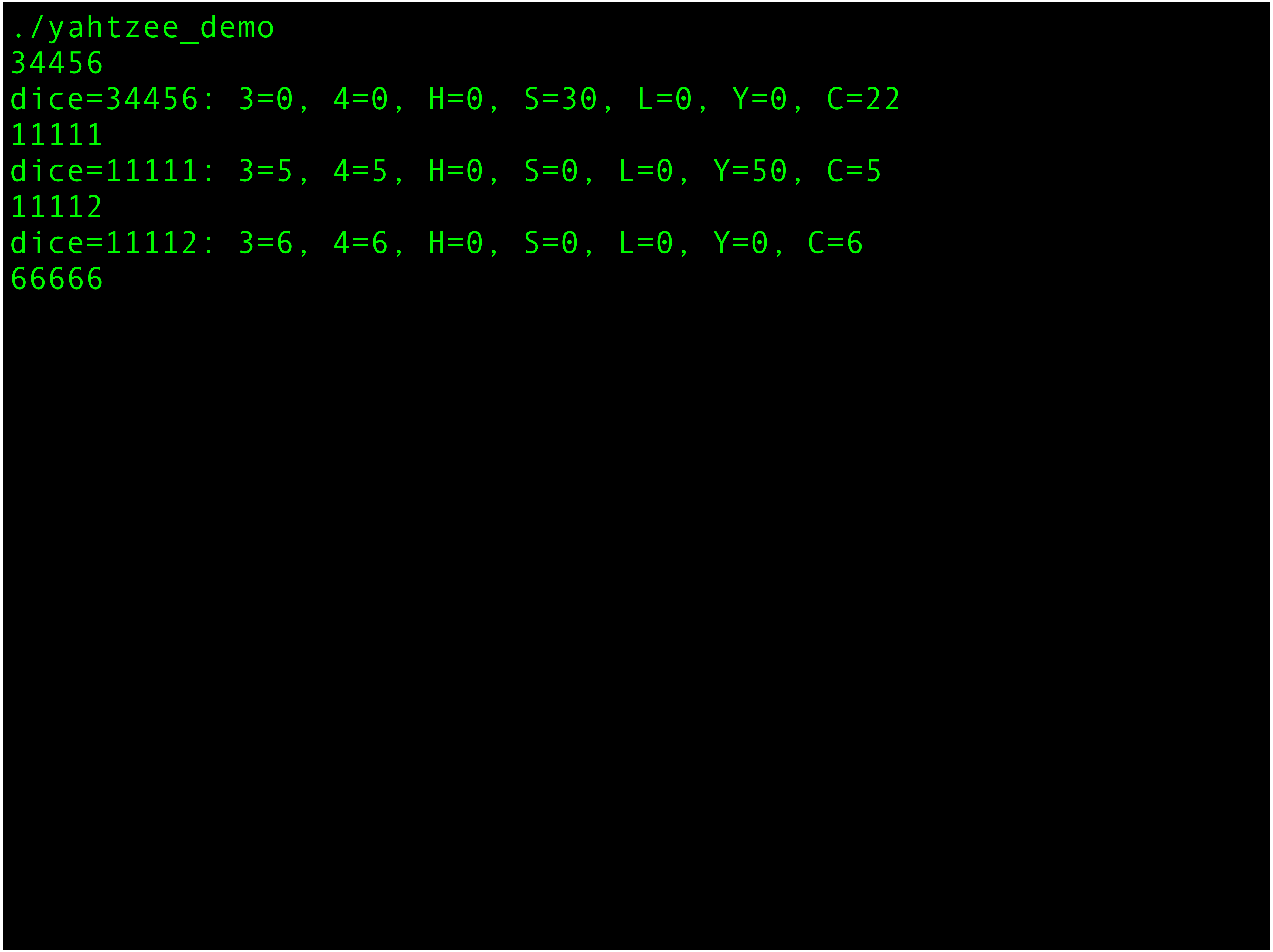
```
./yahtzee_demo
34456
dice=34456: 3=0, 4=0, H=0, S=30, L=0, Y=0, C=22
11111
dice=11111: 3=5, 4=5, H=0, S=0, L=0, Y=50, C=5
```



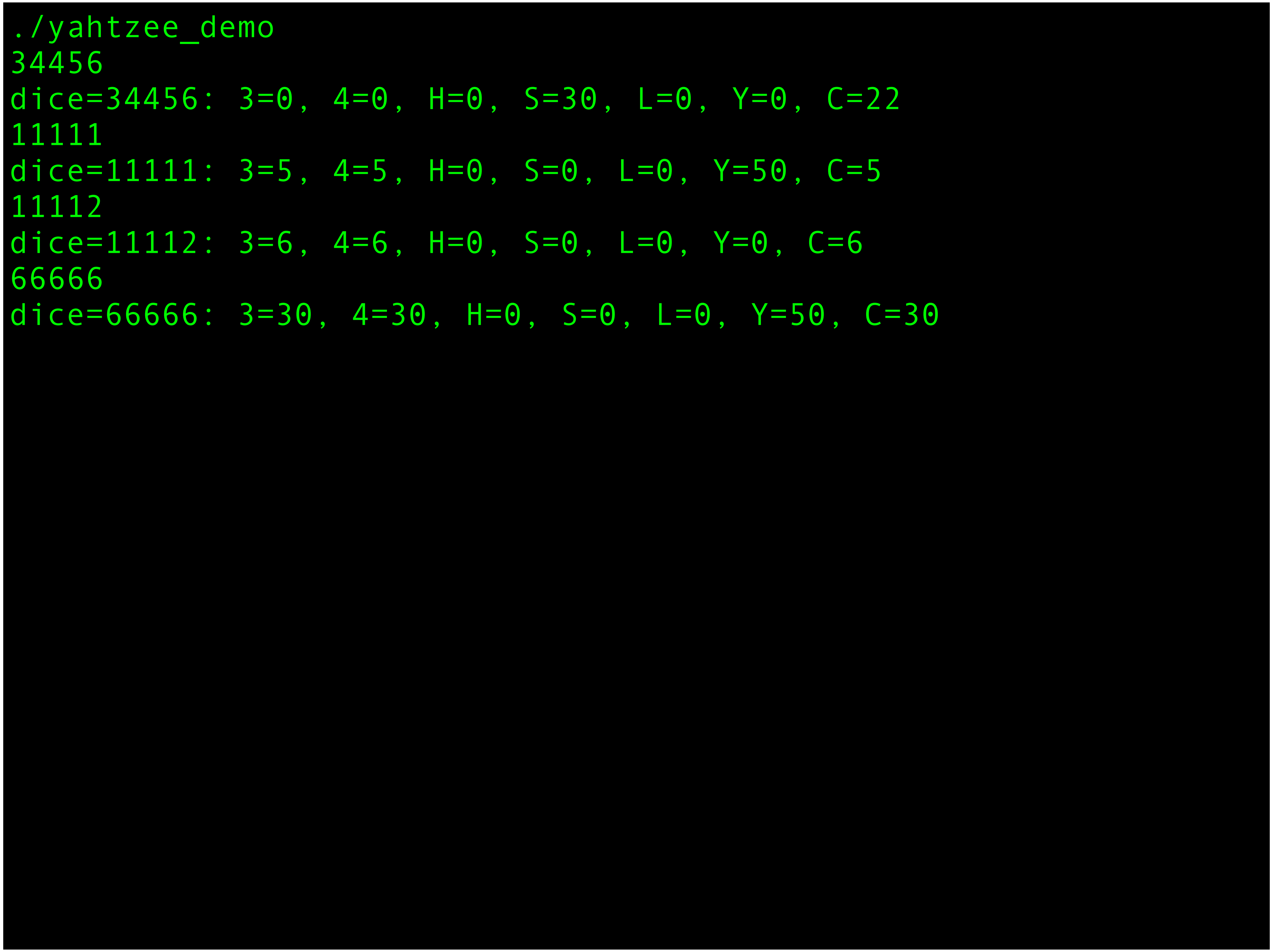
```
./yahtzee_demo
34456
dice=34456: 3=0, 4=0, H=0, S=30, L=0, Y=0, C=22
11111
dice=11111: 3=5, 4=5, H=0, S=0, L=0, Y=50, C=5
11112
```



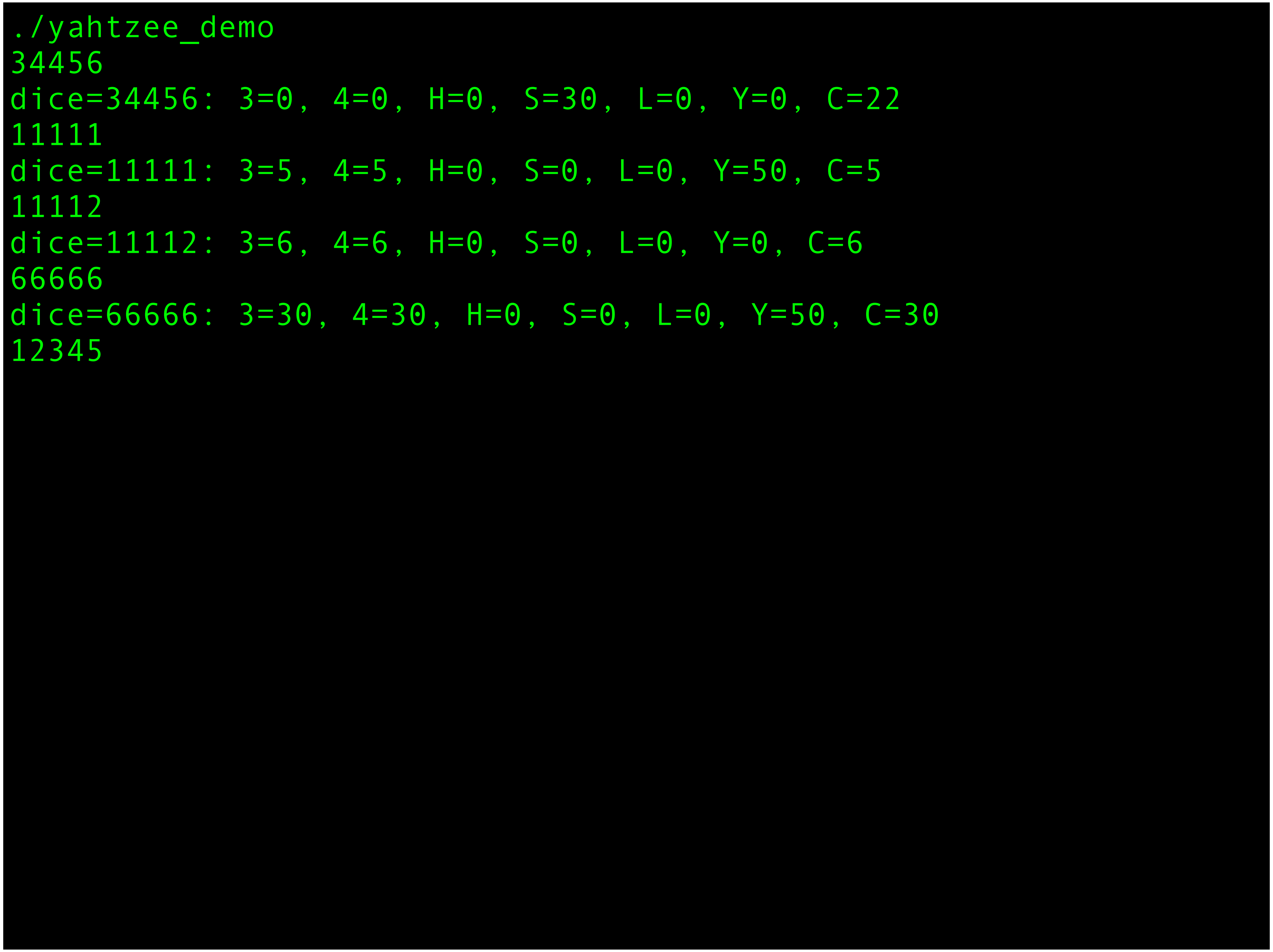
```
./yahtzee_demo
34456
dice=34456: 3=0, 4=0, H=0, S=30, L=0, Y=0, C=22
11111
dice=11111: 3=5, 4=5, H=0, S=0, L=0, Y=50, C=5
11112
dice=11112: 3=6, 4=6, H=0, S=0, L=0, Y=0, C=6
```



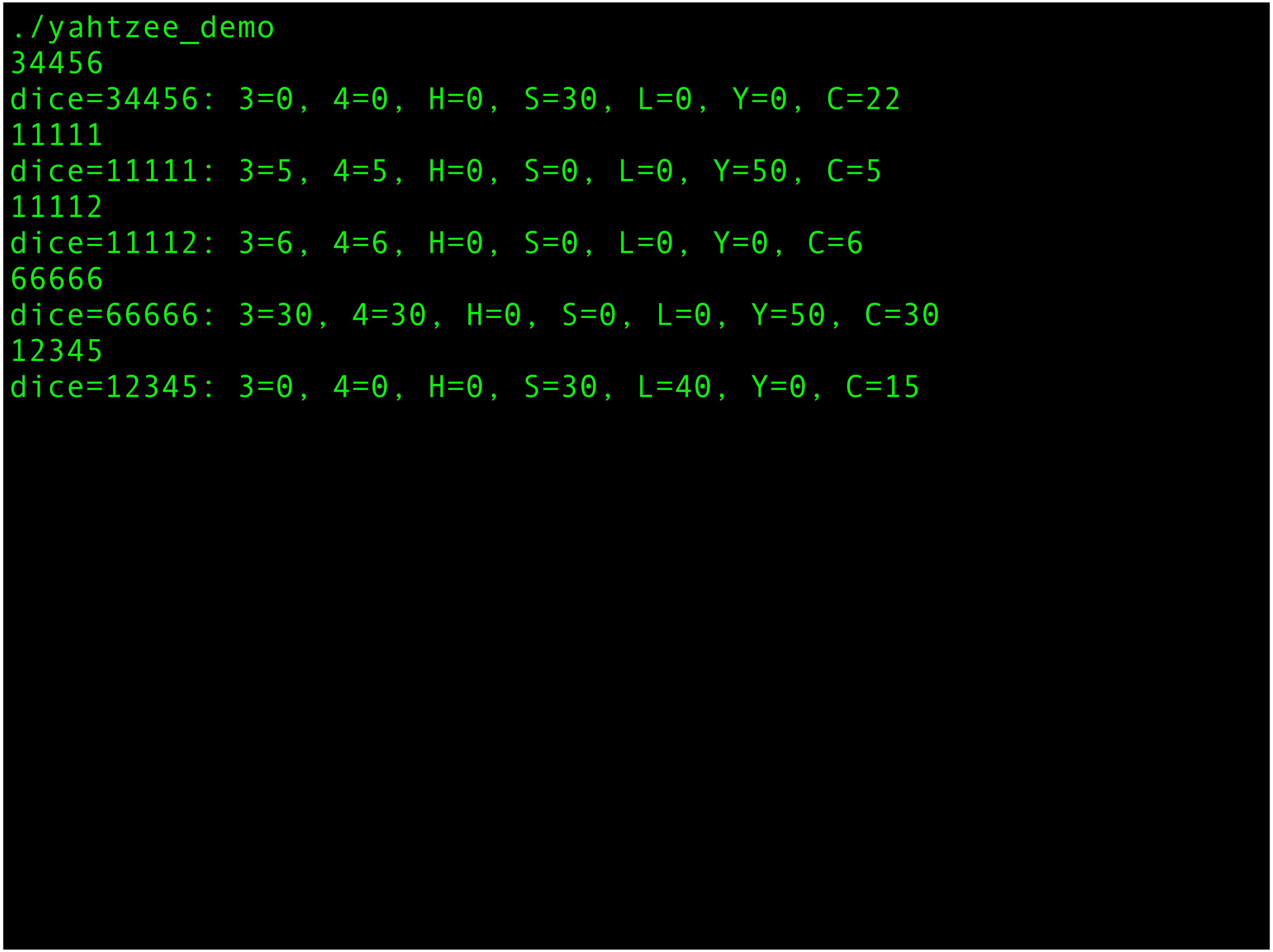
```
./yahtzee_demo
34456
dice=34456: 3=0, 4=0, H=0, S=30, L=0, Y=0, C=22
11111
dice=11111: 3=5, 4=5, H=0, S=0, L=0, Y=50, C=5
11112
dice=11112: 3=6, 4=6, H=0, S=0, L=0, Y=0, C=6
66666
```



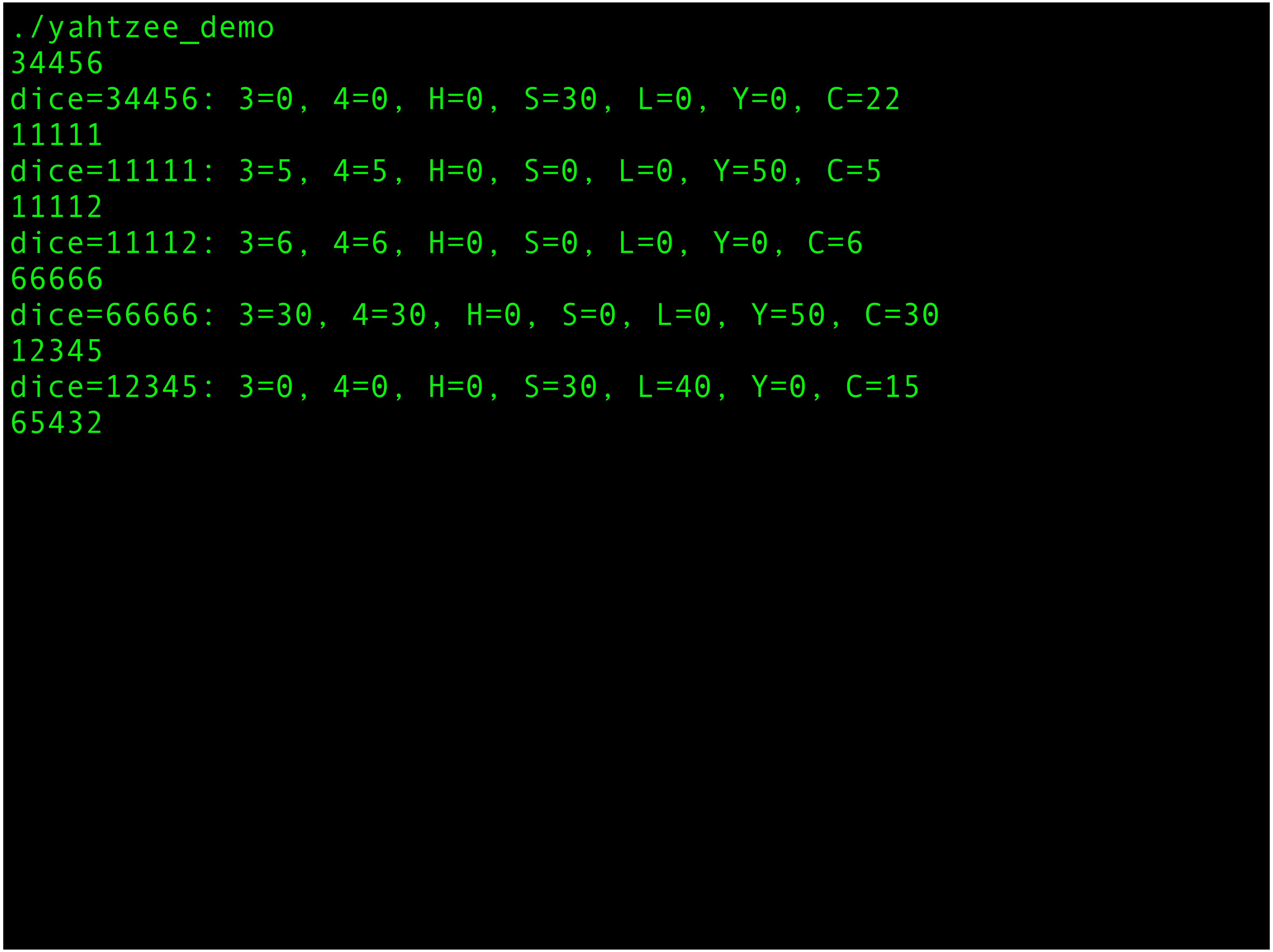
```
./yahtzee_demo
34456
dice=34456: 3=0, 4=0, H=0, S=30, L=0, Y=0, C=22
11111
dice=11111: 3=5, 4=5, H=0, S=0, L=0, Y=50, C=5
11112
dice=11112: 3=6, 4=6, H=0, S=0, L=0, Y=0, C=6
66666
dice=66666: 3=30, 4=30, H=0, S=0, L=0, Y=50, C=30
```



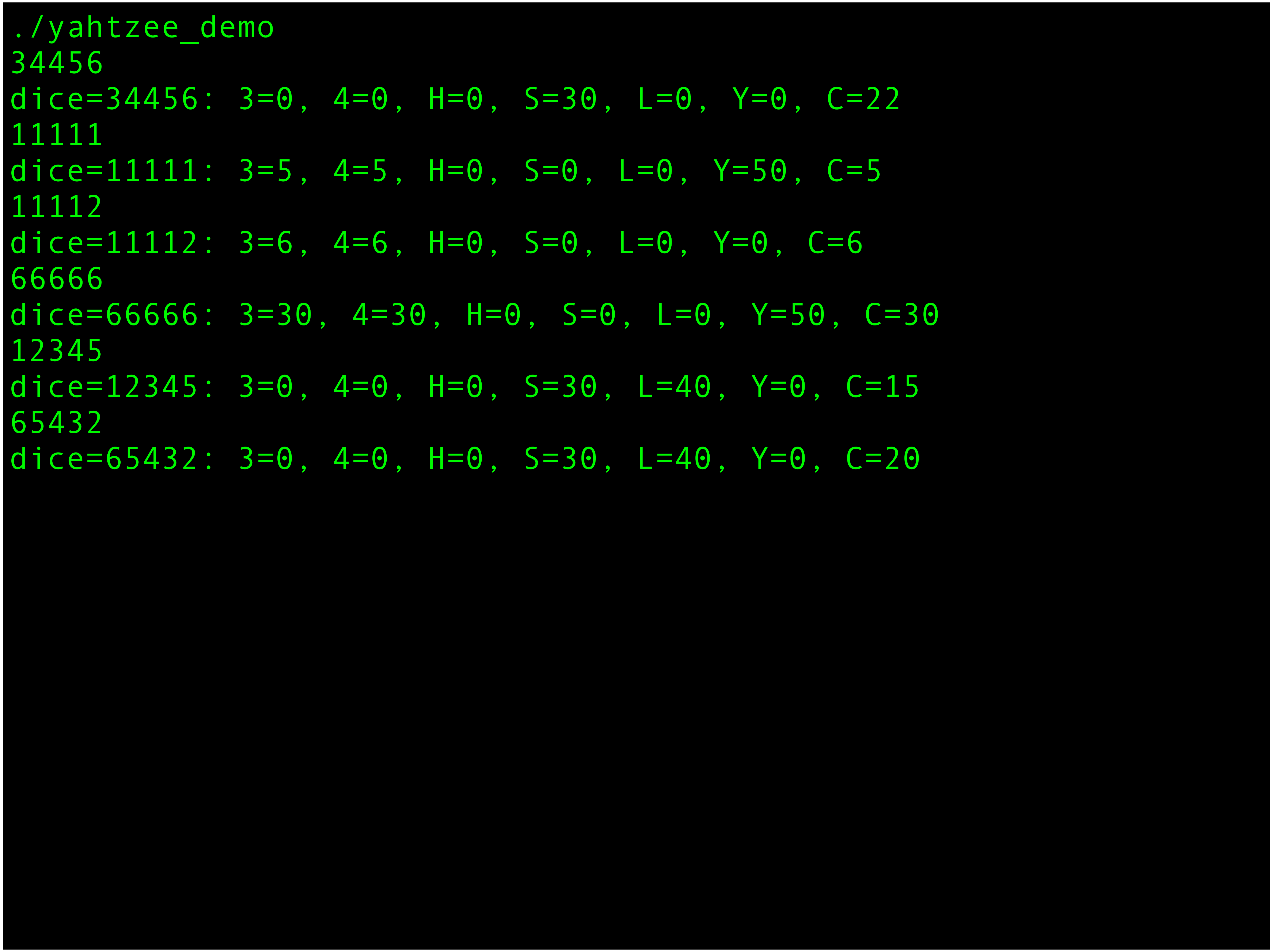
```
./yahtzee_demo
34456
dice=34456: 3=0, 4=0, H=0, S=30, L=0, Y=0, C=22
11111
dice=11111: 3=5, 4=5, H=0, S=0, L=0, Y=50, C=5
11112
dice=11112: 3=6, 4=6, H=0, S=0, L=0, Y=0, C=6
66666
dice=66666: 3=30, 4=30, H=0, S=0, L=0, Y=50, C=30
12345
```



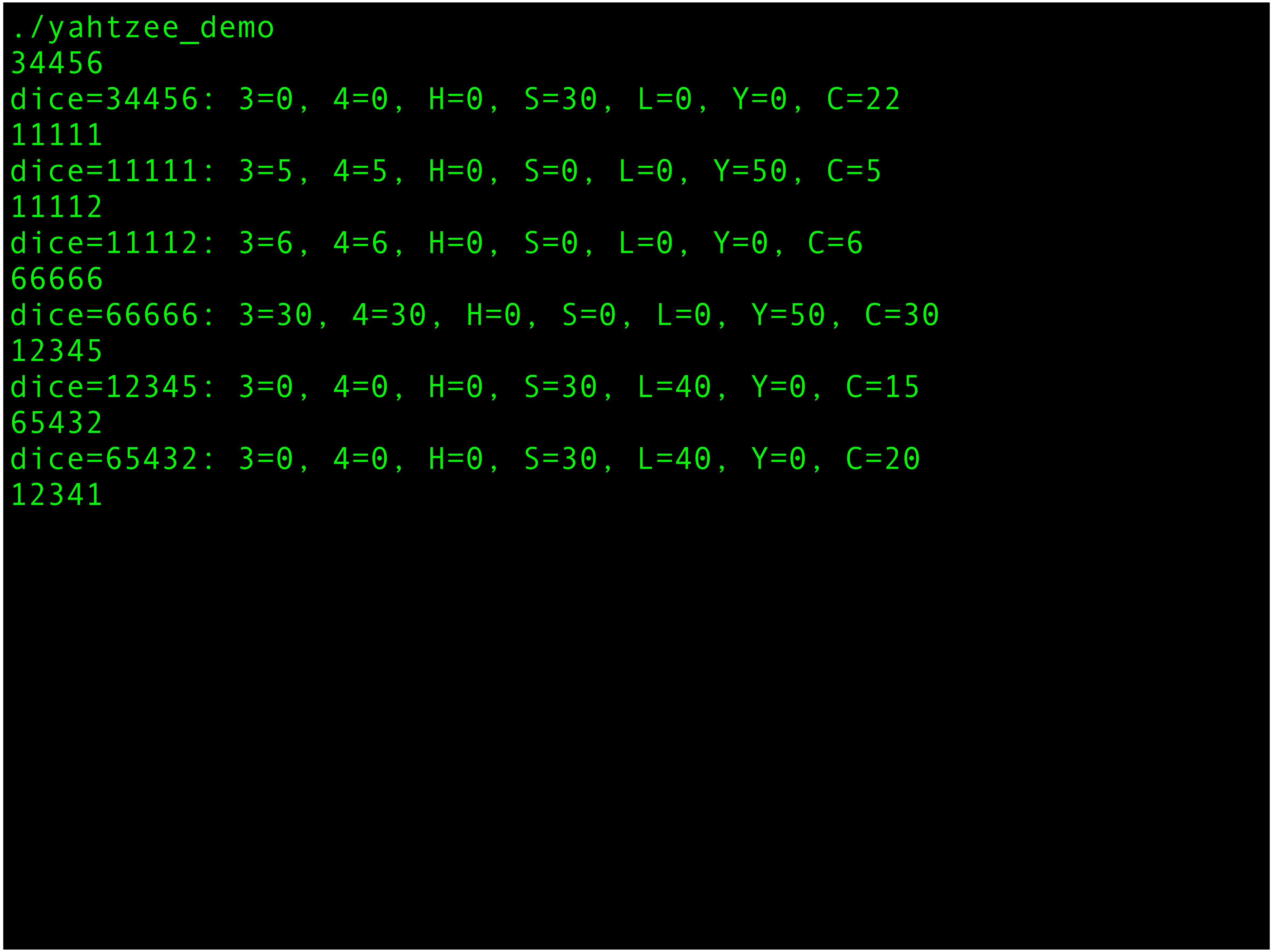
```
./yahtzee_demo
34456
dice=34456: 3=0, 4=0, H=0, S=30, L=0, Y=0, C=22
11111
dice=11111: 3=5, 4=5, H=0, S=0, L=0, Y=50, C=5
11112
dice=11112: 3=6, 4=6, H=0, S=0, L=0, Y=0, C=6
66666
dice=66666: 3=30, 4=30, H=0, S=0, L=0, Y=50, C=30
12345
dice=12345: 3=0, 4=0, H=0, S=30, L=40, Y=0, C=15
```



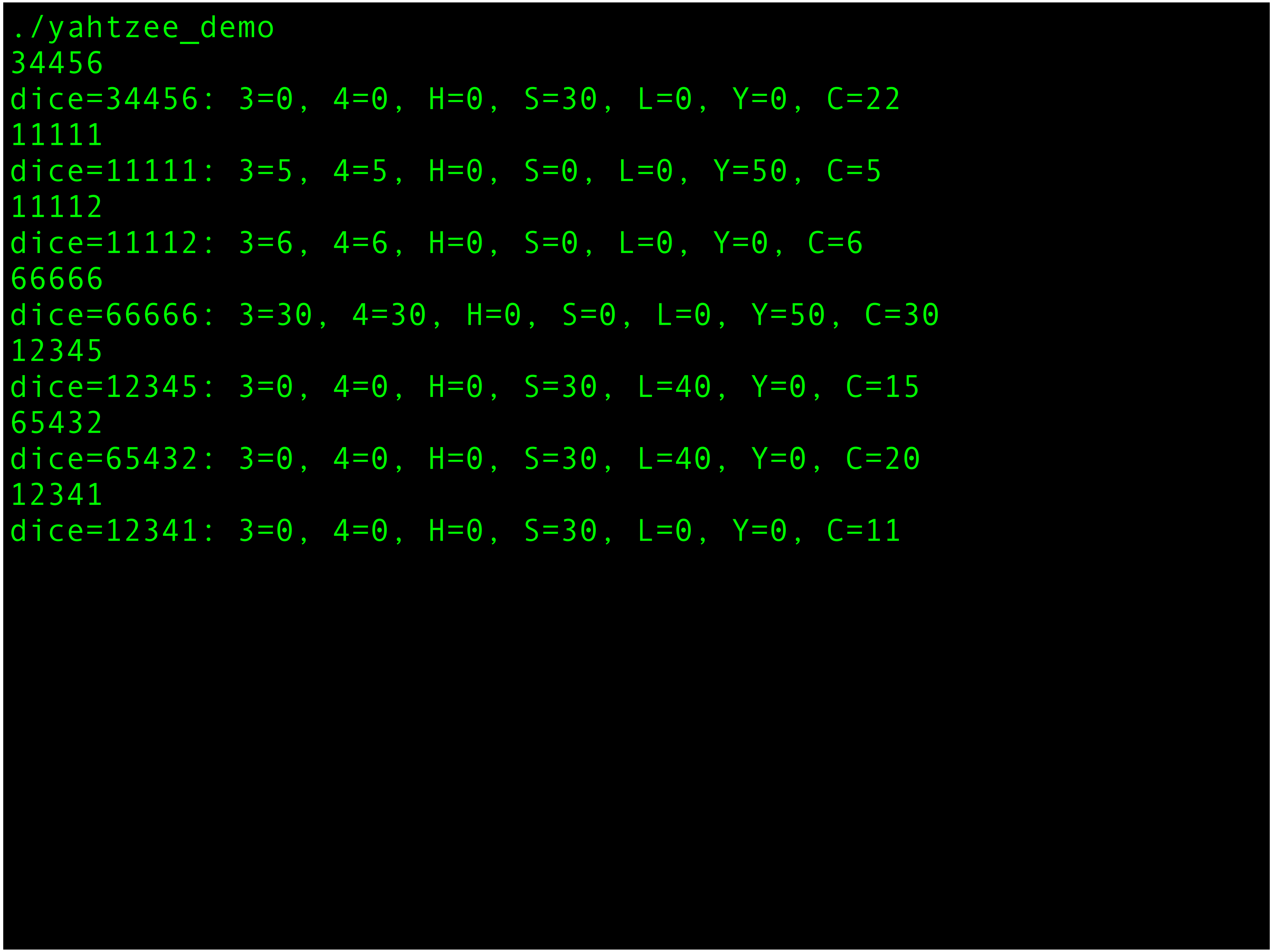
```
./yahtzee_demo
34456
dice=34456: 3=0, 4=0, H=0, S=30, L=0, Y=0, C=22
11111
dice=11111: 3=5, 4=5, H=0, S=0, L=0, Y=50, C=5
11112
dice=11112: 3=6, 4=6, H=0, S=0, L=0, Y=0, C=6
66666
dice=66666: 3=30, 4=30, H=0, S=0, L=0, Y=50, C=30
12345
dice=12345: 3=0, 4=0, H=0, S=30, L=40, Y=0, C=15
65432
```



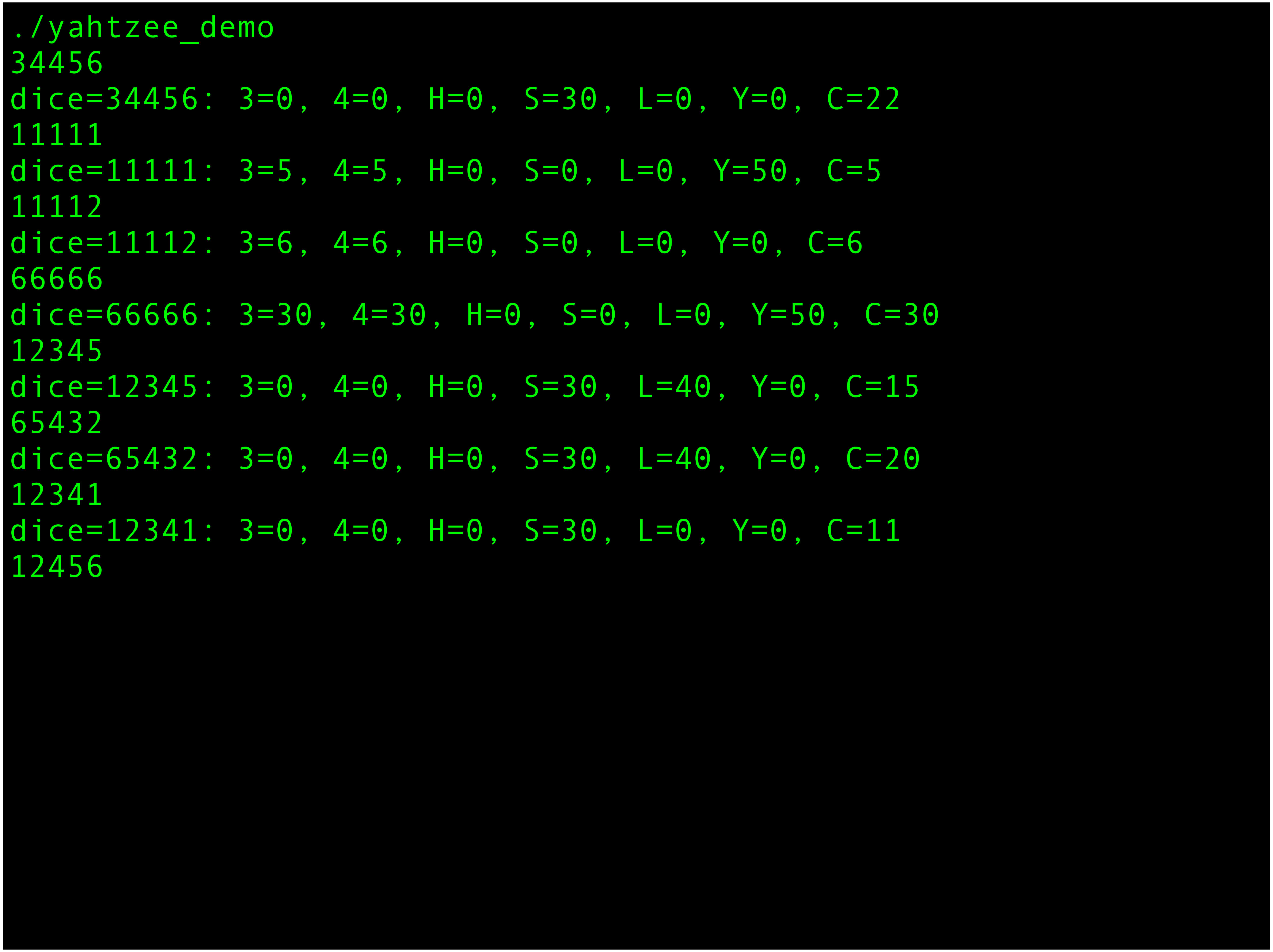
```
./yahtzee_demo
34456
dice=34456: 3=0, 4=0, H=0, S=30, L=0, Y=0, C=22
11111
dice=11111: 3=5, 4=5, H=0, S=0, L=0, Y=50, C=5
11112
dice=11112: 3=6, 4=6, H=0, S=0, L=0, Y=0, C=6
66666
dice=66666: 3=30, 4=30, H=0, S=0, L=0, Y=50, C=30
12345
dice=12345: 3=0, 4=0, H=0, S=30, L=40, Y=0, C=15
65432
dice=65432: 3=0, 4=0, H=0, S=30, L=40, Y=0, C=20
```



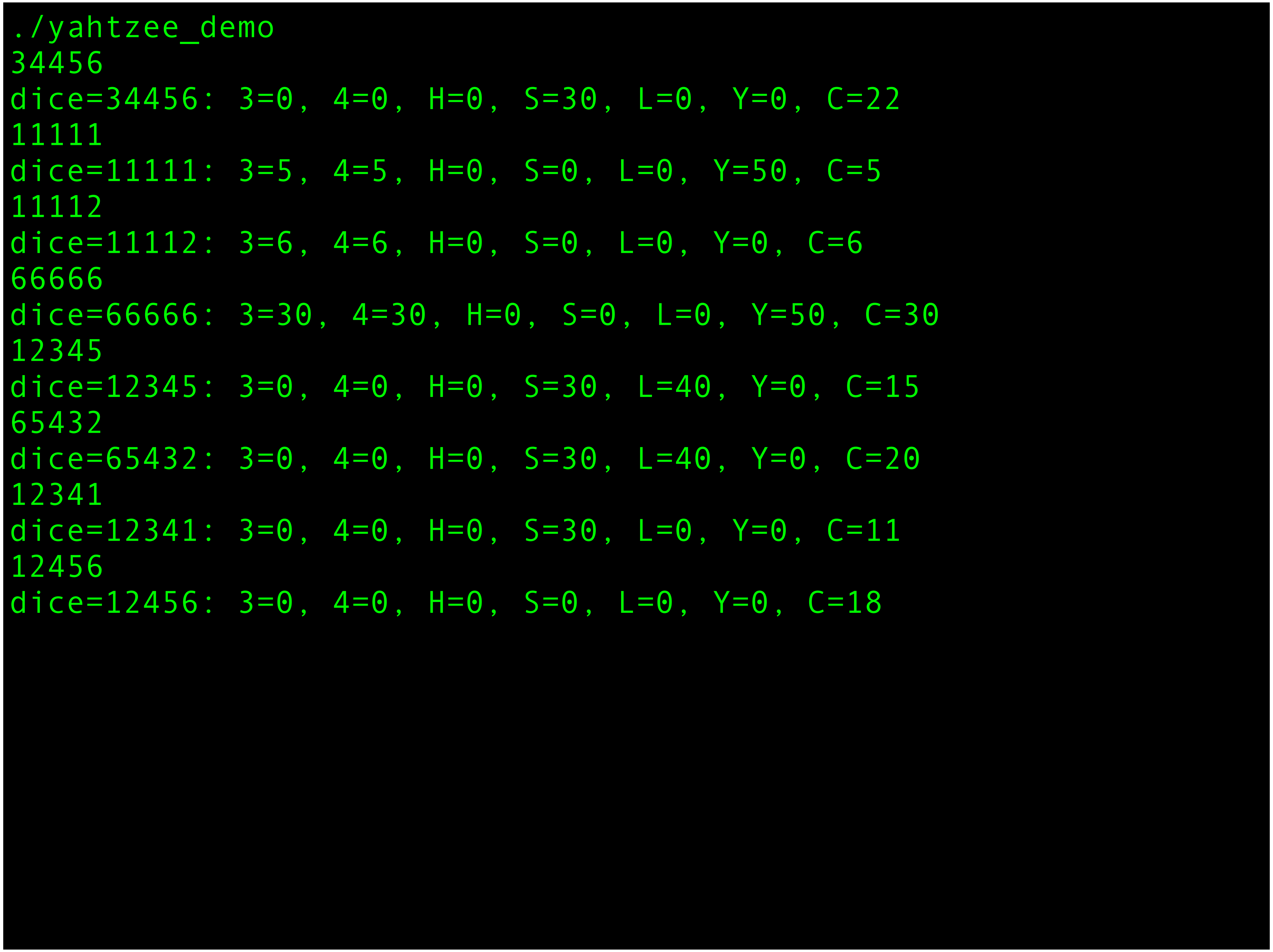
```
./yahtzee_demo
34456
dice=34456: 3=0, 4=0, H=0, S=30, L=0, Y=0, C=22
11111
dice=11111: 3=5, 4=5, H=0, S=0, L=0, Y=50, C=5
11112
dice=11112: 3=6, 4=6, H=0, S=0, L=0, Y=0, C=6
66666
dice=66666: 3=30, 4=30, H=0, S=0, L=0, Y=50, C=30
12345
dice=12345: 3=0, 4=0, H=0, S=30, L=40, Y=0, C=15
65432
dice=65432: 3=0, 4=0, H=0, S=30, L=40, Y=0, C=20
12341
```



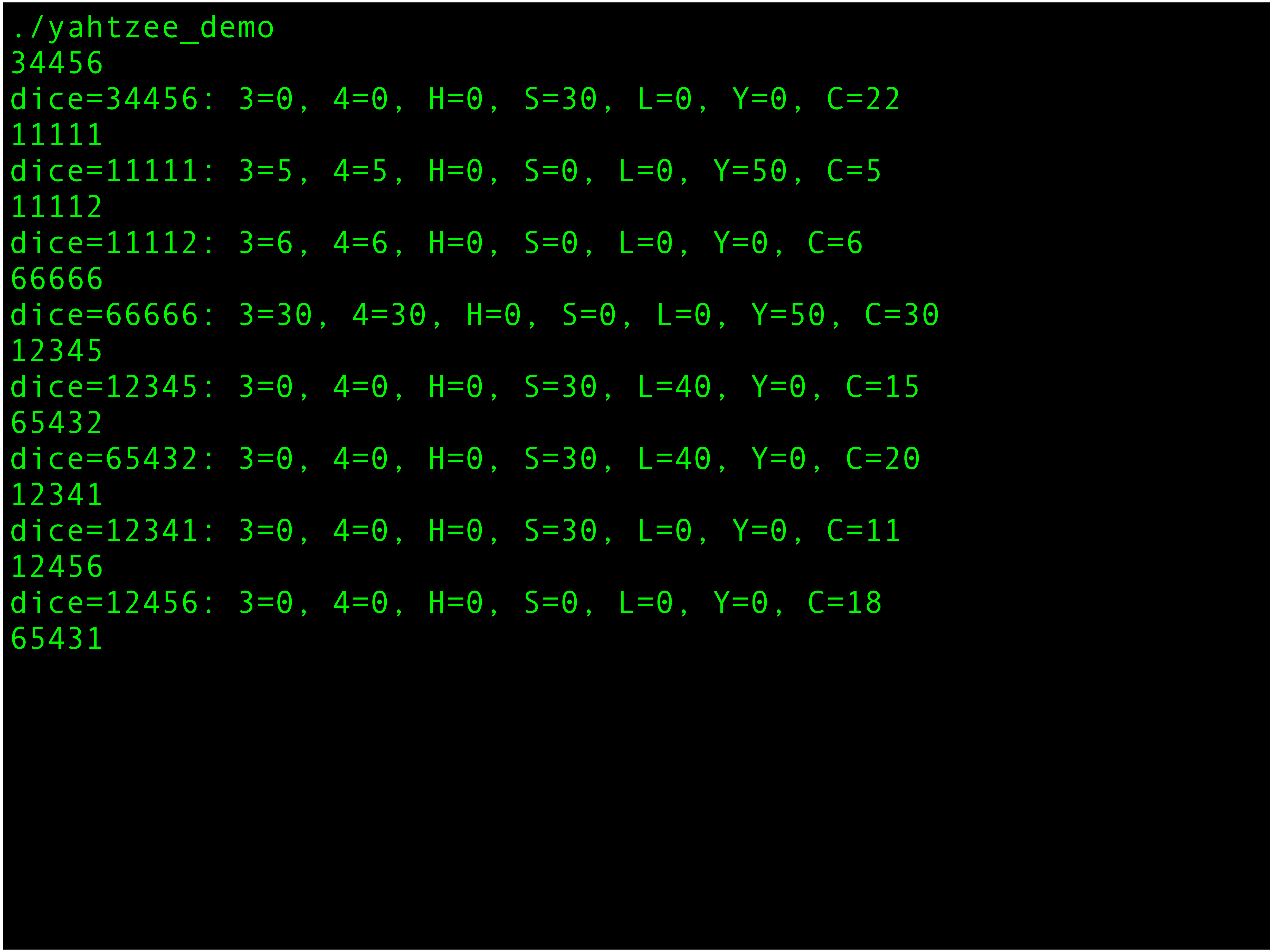
```
./yahtzee_demo
34456
dice=34456: 3=0, 4=0, H=0, S=30, L=0, Y=0, C=22
11111
dice=11111: 3=5, 4=5, H=0, S=0, L=0, Y=50, C=5
11112
dice=11112: 3=6, 4=6, H=0, S=0, L=0, Y=0, C=6
66666
dice=66666: 3=30, 4=30, H=0, S=0, L=0, Y=50, C=30
12345
dice=12345: 3=0, 4=0, H=0, S=30, L=40, Y=0, C=15
65432
dice=65432: 3=0, 4=0, H=0, S=30, L=40, Y=0, C=20
12341
dice=12341: 3=0, 4=0, H=0, S=30, L=0, Y=0, C=11
```



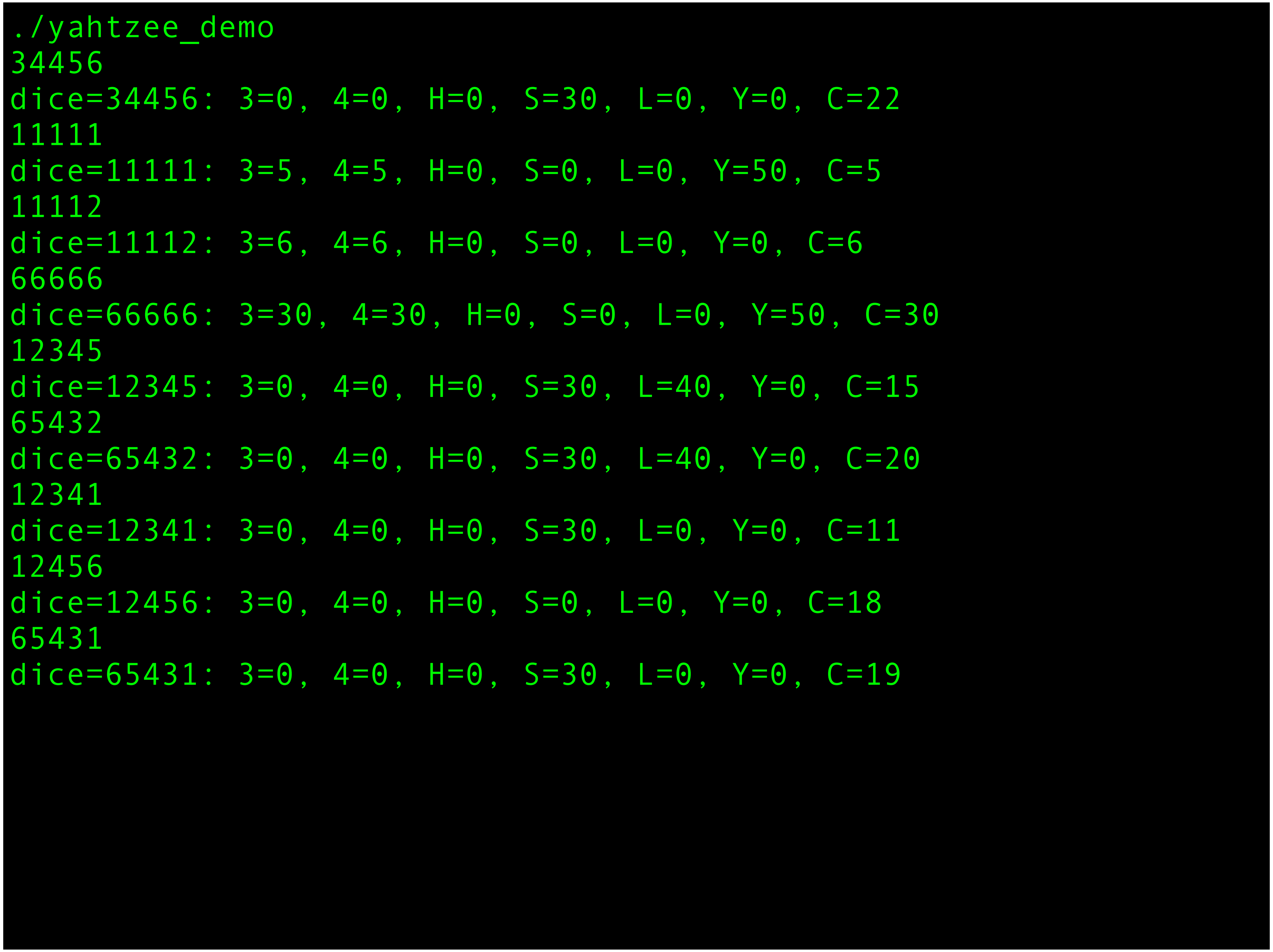
```
./yahtzee_demo
34456
dice=34456: 3=0, 4=0, H=0, S=30, L=0, Y=0, C=22
11111
dice=11111: 3=5, 4=5, H=0, S=0, L=0, Y=50, C=5
11112
dice=11112: 3=6, 4=6, H=0, S=0, L=0, Y=0, C=6
66666
dice=66666: 3=30, 4=30, H=0, S=0, L=0, Y=50, C=30
12345
dice=12345: 3=0, 4=0, H=0, S=30, L=40, Y=0, C=15
65432
dice=65432: 3=0, 4=0, H=0, S=30, L=40, Y=0, C=20
12341
dice=12341: 3=0, 4=0, H=0, S=30, L=0, Y=0, C=11
12456
```



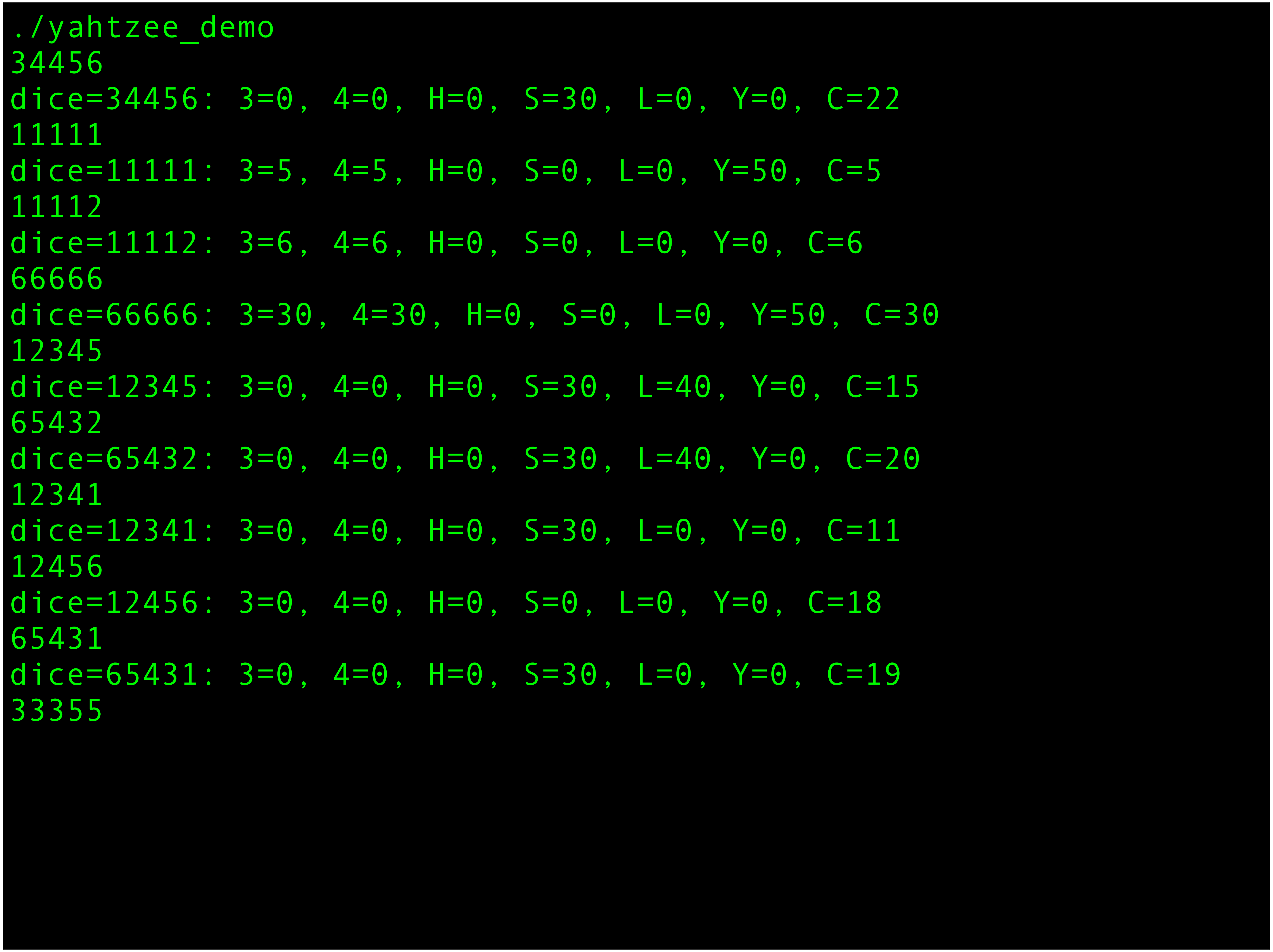
```
./yahtzee_demo
34456
dice=34456: 3=0, 4=0, H=0, S=30, L=0, Y=0, C=22
11111
dice=11111: 3=5, 4=5, H=0, S=0, L=0, Y=50, C=5
11112
dice=11112: 3=6, 4=6, H=0, S=0, L=0, Y=0, C=6
66666
dice=66666: 3=30, 4=30, H=0, S=0, L=0, Y=50, C=30
12345
dice=12345: 3=0, 4=0, H=0, S=30, L=40, Y=0, C=15
65432
dice=65432: 3=0, 4=0, H=0, S=30, L=40, Y=0, C=20
12341
dice=12341: 3=0, 4=0, H=0, S=30, L=0, Y=0, C=11
12456
dice=12456: 3=0, 4=0, H=0, S=0, L=0, Y=0, C=18
```



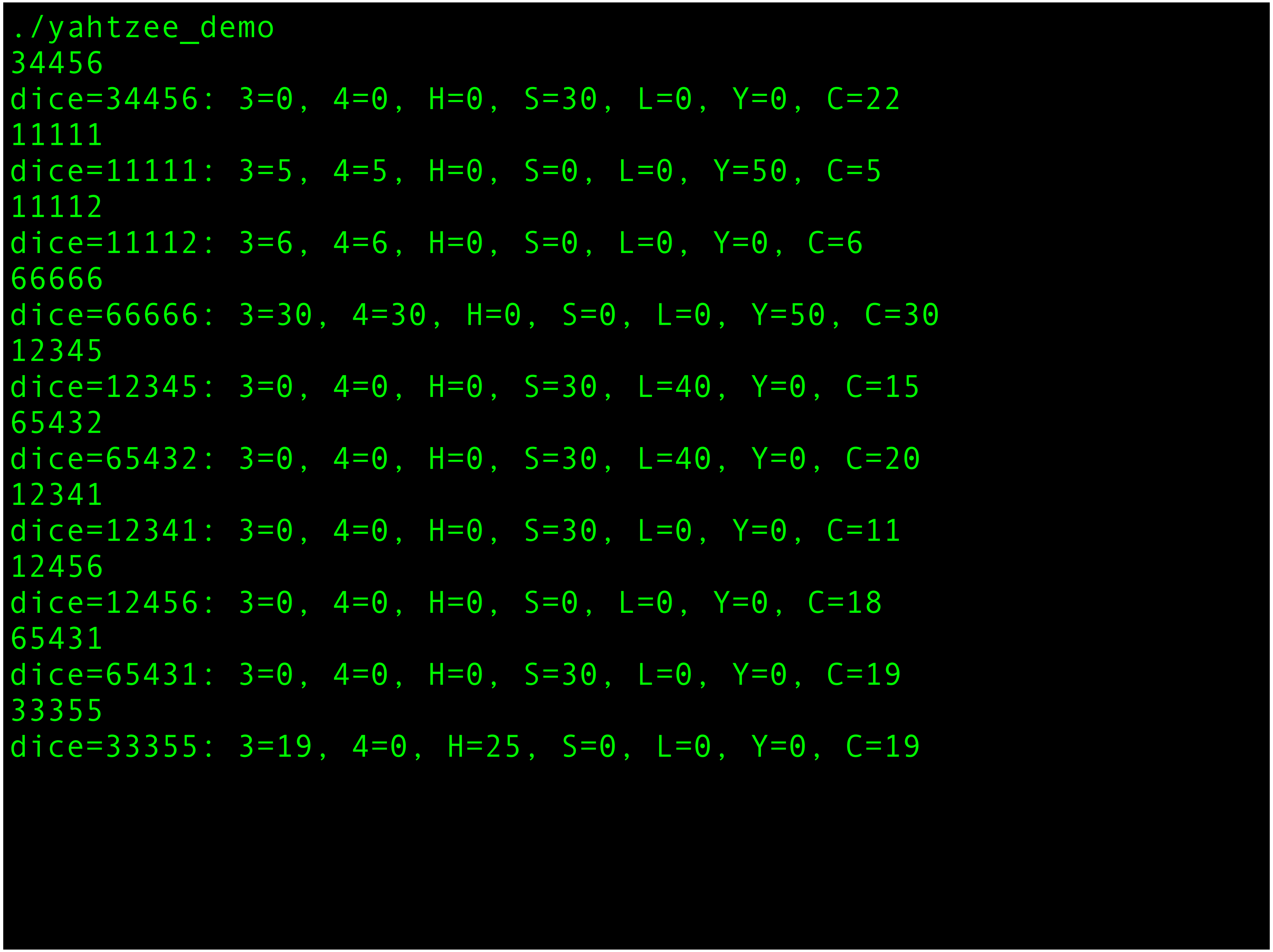
```
./yahtzee_demo
34456
dice=34456: 3=0, 4=0, H=0, S=30, L=0, Y=0, C=22
11111
dice=11111: 3=5, 4=5, H=0, S=0, L=0, Y=50, C=5
11112
dice=11112: 3=6, 4=6, H=0, S=0, L=0, Y=0, C=6
66666
dice=66666: 3=30, 4=30, H=0, S=0, L=0, Y=50, C=30
12345
dice=12345: 3=0, 4=0, H=0, S=30, L=40, Y=0, C=15
65432
dice=65432: 3=0, 4=0, H=0, S=30, L=40, Y=0, C=20
12341
dice=12341: 3=0, 4=0, H=0, S=30, L=0, Y=0, C=11
12456
dice=12456: 3=0, 4=0, H=0, S=0, L=0, Y=0, C=18
65431
```



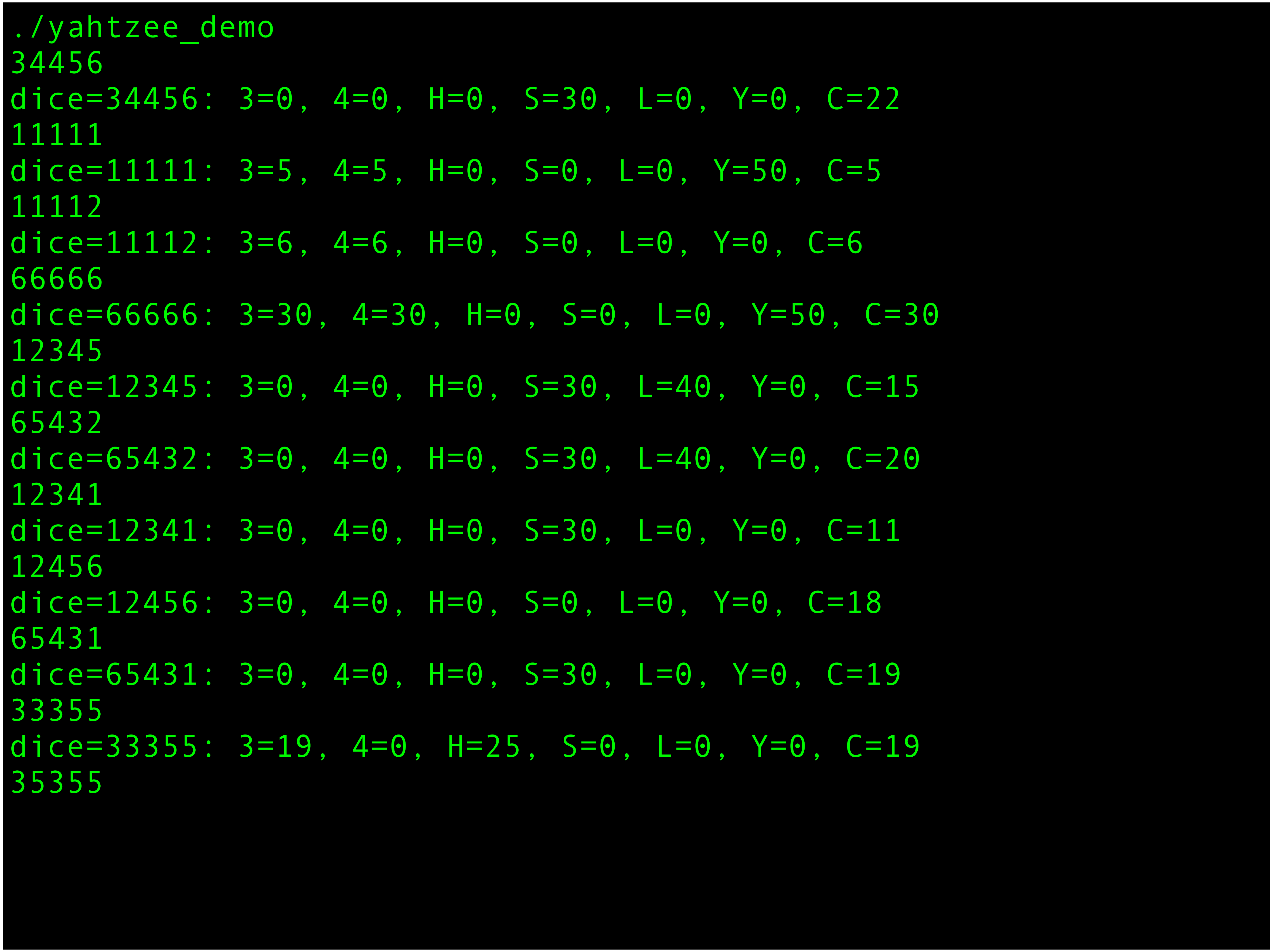
```
./yahtzee_demo
34456
dice=34456: 3=0, 4=0, H=0, S=30, L=0, Y=0, C=22
11111
dice=11111: 3=5, 4=5, H=0, S=0, L=0, Y=50, C=5
11112
dice=11112: 3=6, 4=6, H=0, S=0, L=0, Y=0, C=6
66666
dice=66666: 3=30, 4=30, H=0, S=0, L=0, Y=50, C=30
12345
dice=12345: 3=0, 4=0, H=0, S=30, L=40, Y=0, C=15
65432
dice=65432: 3=0, 4=0, H=0, S=30, L=40, Y=0, C=20
12341
dice=12341: 3=0, 4=0, H=0, S=30, L=0, Y=0, C=11
12456
dice=12456: 3=0, 4=0, H=0, S=0, L=0, Y=0, C=18
65431
dice=65431: 3=0, 4=0, H=0, S=30, L=0, Y=0, C=19
```



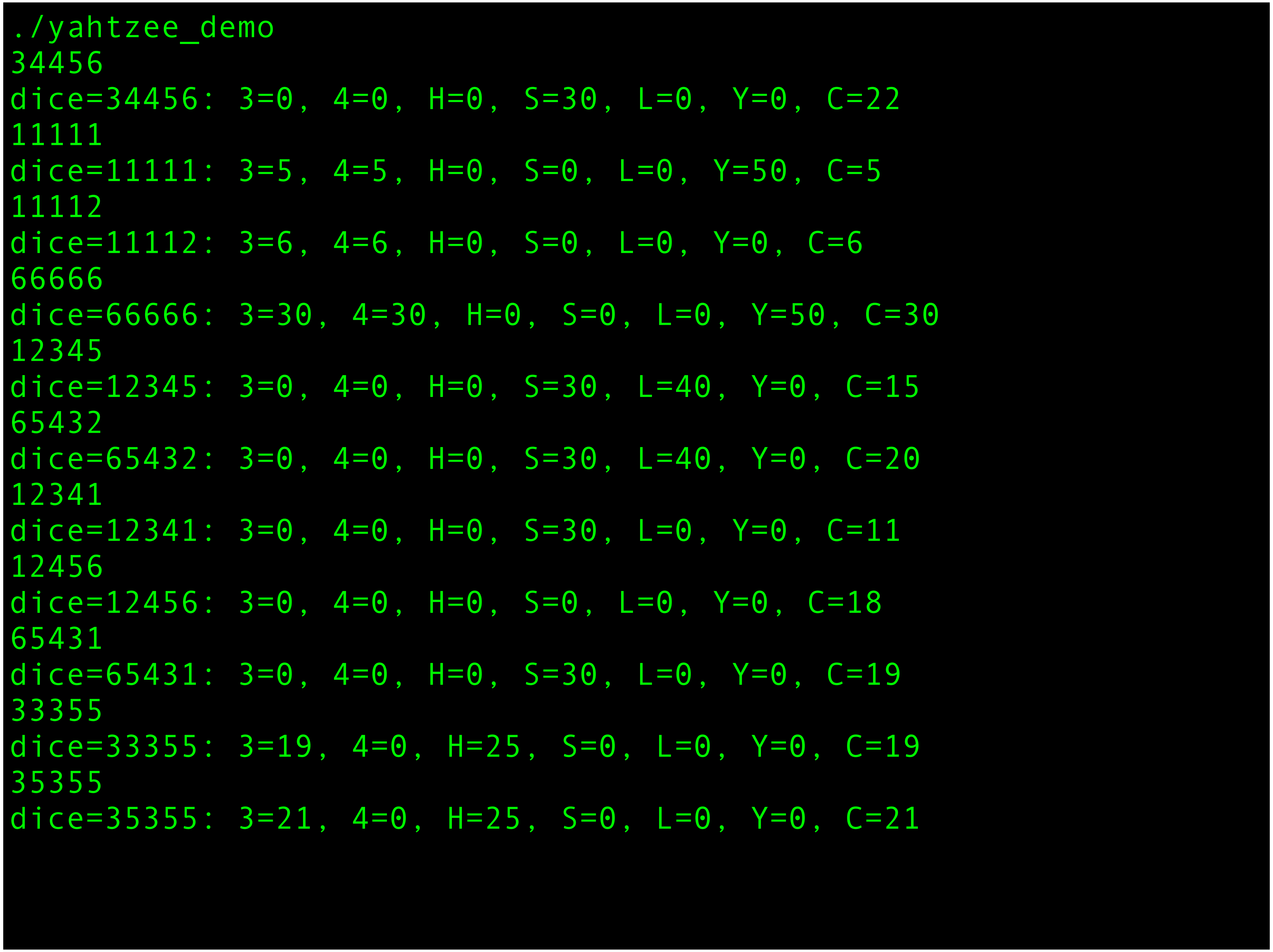
```
./yahtzee_demo
34456
dice=34456: 3=0, 4=0, H=0, S=30, L=0, Y=0, C=22
11111
dice=11111: 3=5, 4=5, H=0, S=0, L=0, Y=50, C=5
11112
dice=11112: 3=6, 4=6, H=0, S=0, L=0, Y=0, C=6
66666
dice=66666: 3=30, 4=30, H=0, S=0, L=0, Y=50, C=30
12345
dice=12345: 3=0, 4=0, H=0, S=30, L=40, Y=0, C=15
65432
dice=65432: 3=0, 4=0, H=0, S=30, L=40, Y=0, C=20
12341
dice=12341: 3=0, 4=0, H=0, S=30, L=0, Y=0, C=11
12456
dice=12456: 3=0, 4=0, H=0, S=0, L=0, Y=0, C=18
65431
dice=65431: 3=0, 4=0, H=0, S=30, L=0, Y=0, C=19
33355
```



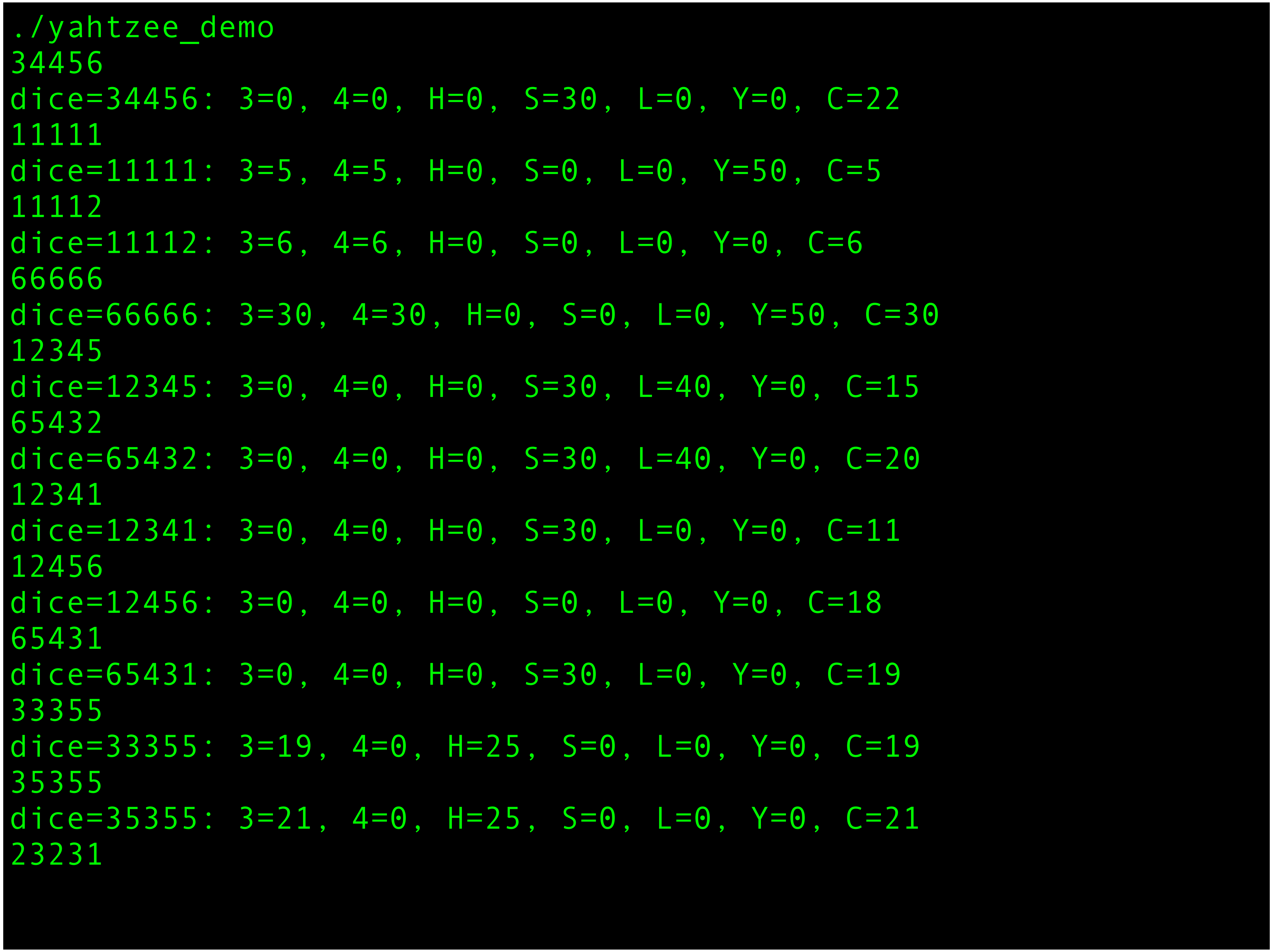
```
./yahtzee_demo
34456
dice=34456: 3=0, 4=0, H=0, S=30, L=0, Y=0, C=22
11111
dice=11111: 3=5, 4=5, H=0, S=0, L=0, Y=50, C=5
11112
dice=11112: 3=6, 4=6, H=0, S=0, L=0, Y=0, C=6
66666
dice=66666: 3=30, 4=30, H=0, S=0, L=0, Y=50, C=30
12345
dice=12345: 3=0, 4=0, H=0, S=30, L=40, Y=0, C=15
65432
dice=65432: 3=0, 4=0, H=0, S=30, L=40, Y=0, C=20
12341
dice=12341: 3=0, 4=0, H=0, S=30, L=0, Y=0, C=11
12456
dice=12456: 3=0, 4=0, H=0, S=0, L=0, Y=0, C=18
65431
dice=65431: 3=0, 4=0, H=0, S=30, L=0, Y=0, C=19
33355
dice=33355: 3=19, 4=0, H=25, S=0, L=0, Y=0, C=19
```

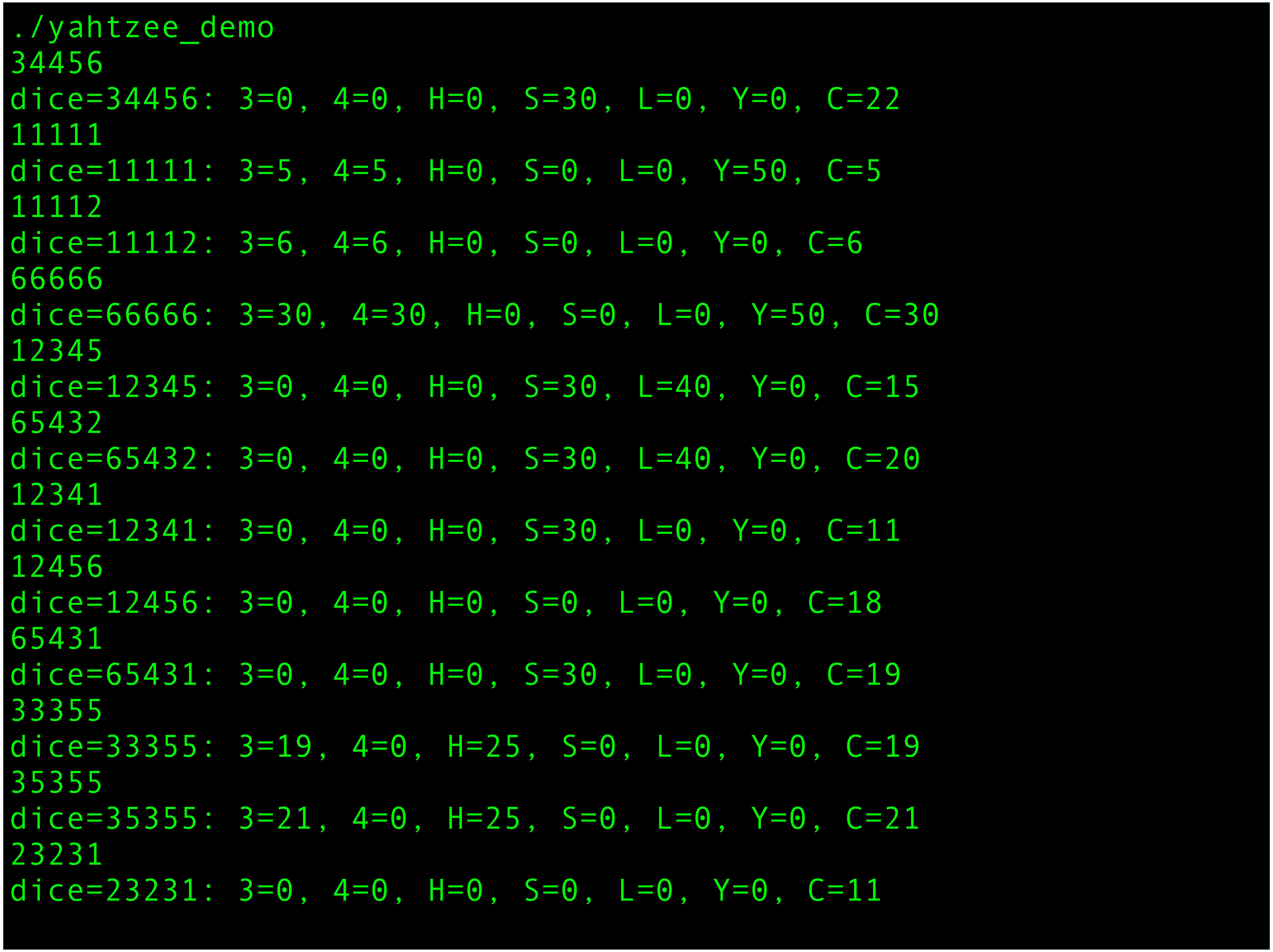


```
./yahtzee_demo
34456
dice=34456: 3=0, 4=0, H=0, S=30, L=0, Y=0, C=22
11111
dice=11111: 3=5, 4=5, H=0, S=0, L=0, Y=50, C=5
11112
dice=11112: 3=6, 4=6, H=0, S=0, L=0, Y=0, C=6
66666
dice=66666: 3=30, 4=30, H=0, S=0, L=0, Y=50, C=30
12345
dice=12345: 3=0, 4=0, H=0, S=30, L=40, Y=0, C=15
65432
dice=65432: 3=0, 4=0, H=0, S=30, L=40, Y=0, C=20
12341
dice=12341: 3=0, 4=0, H=0, S=30, L=0, Y=0, C=11
12456
dice=12456: 3=0, 4=0, H=0, S=0, L=0, Y=0, C=18
65431
dice=65431: 3=0, 4=0, H=0, S=30, L=0, Y=0, C=19
33355
dice=33355: 3=19, 4=0, H=25, S=0, L=0, Y=0, C=19
35355
```

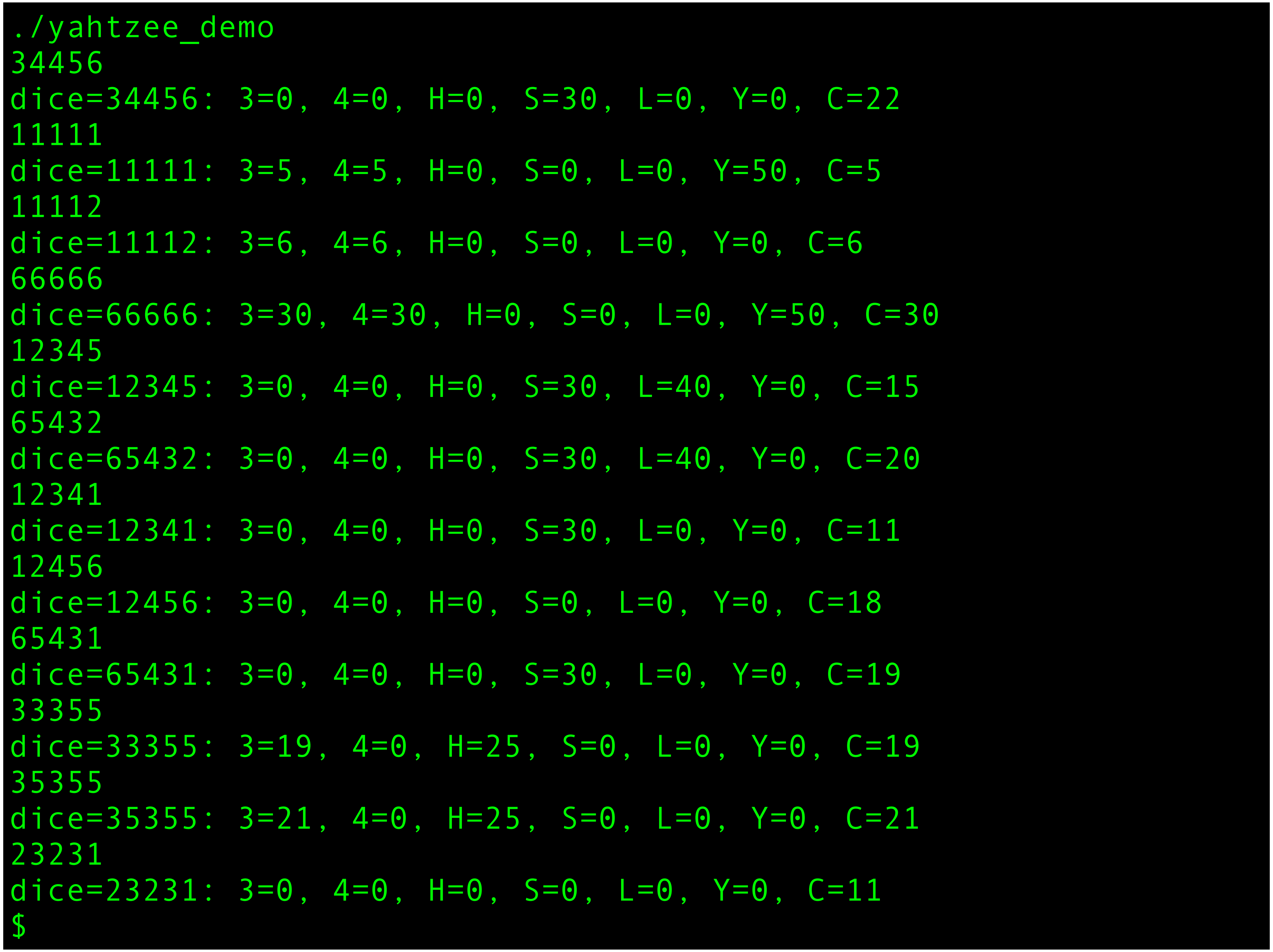


```
./yahtzee_demo
34456
dice=34456: 3=0, 4=0, H=0, S=30, L=0, Y=0, C=22
11111
dice=11111: 3=5, 4=5, H=0, S=0, L=0, Y=50, C=5
11112
dice=11112: 3=6, 4=6, H=0, S=0, L=0, Y=0, C=6
66666
dice=66666: 3=30, 4=30, H=0, S=0, L=0, Y=50, C=30
12345
dice=12345: 3=0, 4=0, H=0, S=30, L=40, Y=0, C=15
65432
dice=65432: 3=0, 4=0, H=0, S=30, L=40, Y=0, C=20
12341
dice=12341: 3=0, 4=0, H=0, S=30, L=0, Y=0, C=11
12456
dice=12456: 3=0, 4=0, H=0, S=0, L=0, Y=0, C=18
65431
dice=65431: 3=0, 4=0, H=0, S=30, L=0, Y=0, C=19
33355
dice=33355: 3=19, 4=0, H=25, S=0, L=0, Y=0, C=19
35355
dice=35355: 3=21, 4=0, H=25, S=0, L=0, Y=0, C=21
```


```
./yahtzee_demo
34456
dice=34456: 3=0, 4=0, H=0, S=30, L=0, Y=0, C=22
11111
dice=11111: 3=5, 4=5, H=0, S=0, L=0, Y=50, C=5
11112
dice=11112: 3=6, 4=6, H=0, S=0, L=0, Y=0, C=6
66666
dice=66666: 3=30, 4=30, H=0, S=0, L=0, Y=50, C=30
12345
dice=12345: 3=0, 4=0, H=0, S=30, L=40, Y=0, C=15
65432
dice=65432: 3=0, 4=0, H=0, S=30, L=40, Y=0, C=20
12341
dice=12341: 3=0, 4=0, H=0, S=30, L=0, Y=0, C=11
12456
dice=12456: 3=0, 4=0, H=0, S=0, L=0, Y=0, C=18
65431
dice=65431: 3=0, 4=0, H=0, S=30, L=0, Y=0, C=19
33355
dice=33355: 3=19, 4=0, H=25, S=0, L=0, Y=0, C=19
35355
dice=35355: 3=21, 4=0, H=25, S=0, L=0, Y=0, C=21
23231
```



```
./yahtzee_demo
34456
dice=34456: 3=0, 4=0, H=0, S=30, L=0, Y=0, C=22
11111
dice=11111: 3=5, 4=5, H=0, S=0, L=0, Y=50, C=5
11112
dice=11112: 3=6, 4=6, H=0, S=0, L=0, Y=0, C=6
66666
dice=66666: 3=30, 4=30, H=0, S=0, L=0, Y=50, C=30
12345
dice=12345: 3=0, 4=0, H=0, S=30, L=40, Y=0, C=15
65432
dice=65432: 3=0, 4=0, H=0, S=30, L=40, Y=0, C=20
12341
dice=12341: 3=0, 4=0, H=0, S=30, L=0, Y=0, C=11
12456
dice=12456: 3=0, 4=0, H=0, S=0, L=0, Y=0, C=18
65431
dice=65431: 3=0, 4=0, H=0, S=30, L=0, Y=0, C=19
33355
dice=33355: 3=19, 4=0, H=25, S=0, L=0, Y=0, C=19
35355
dice=35355: 3=21, 4=0, H=25, S=0, L=0, Y=0, C=21
23231
dice=23231: 3=0, 4=0, H=0, S=0, L=0, Y=0, C=11
```



```
./yahtzee_demo
34456
dice=34456: 3=0, 4=0, H=0, S=30, L=0, Y=0, C=22
11111
dice=11111: 3=5, 4=5, H=0, S=0, L=0, Y=50, C=5
11112
dice=11112: 3=6, 4=6, H=0, S=0, L=0, Y=0, C=6
66666
dice=66666: 3=30, 4=30, H=0, S=0, L=0, Y=50, C=30
12345
dice=12345: 3=0, 4=0, H=0, S=30, L=40, Y=0, C=15
65432
dice=65432: 3=0, 4=0, H=0, S=30, L=40, Y=0, C=20
12341
dice=12341: 3=0, 4=0, H=0, S=30, L=0, Y=0, C=11
12456
dice=12456: 3=0, 4=0, H=0, S=0, L=0, Y=0, C=18
65431
dice=65431: 3=0, 4=0, H=0, S=30, L=0, Y=0, C=19
33355
dice=33355: 3=19, 4=0, H=25, S=0, L=0, Y=0, C=19
35355
dice=35355: 3=21, 4=0, H=25, S=0, L=0, Y=0, C=21
23231
dice=23231: 3=0, 4=0, H=0, S=0, L=0, Y=0, C=11
$
```

Thursday

Thursday

Looks good! Seems like you are done already... The files you receive will have about a 100000 dice, and you need to calculate the score pretty fast.

Thursday

Looks good! Seems like you are done already... The files you receive will have about a 100000 dice, and you need to calculate the score pretty fast.

100000? Fast? How fast?

Thursday

Looks good! Seems like you are done already... The files you receive will have about a 100000 dice, and you need to calculate the score pretty fast.

100000? Fast? How fast?

can you do it in a few seconds?

Thursday

Looks good! Seems like you are done already... The files you receive will have about a 100000 dice, and you need to calculate the score pretty fast.

100000? Fast? How fast?

can you do it in a few seconds?

eat flaming death!

Thursday

Looks good! Seems like you are done already... The files you receive will have about a 100000 dice, and you need to calculate the score pretty fast.

100000? Fast? How fast?

can you do it in a few seconds?

eat flaming death!

I need it by tomorrow. Right?

Thursday

Looks good! Seems like you are done already... The files you receive will have about a 100000 dice, and you need to calculate the score pretty fast.

100000? Fast? How fast?

can you do it in a few seconds?

eat flaming death!

I need it by tomorrow. Right?

sure

last minute requirement:

must have buffered IO

last minute requirement:

must have buffered IO

- write a char to output buffer

last minute requirement:

must have buffered IO

- write a char to output buffer
- flush output buffer

last minute requirement:

must have buffered IO

- write a char to output buffer
- flush output buffer
- read a char from an input buffer

last minute requirement:

must have buffered IO

- write a char to output buffer
- flush output buffer
- read a char from an input buffer
- fill input buffer

How to use TDD to implement buffered IO?

```
kernel:
        int 0x80
        ret

%define SYS_EXIT 1
%define SYS_READ 3
%define SYS_WRITE 4

%define STDIN 0
%define STDOUT 1
%define STDERR 2

%define EXIT_SUCCESS 0
%define EXIT_FAILURE 1

%macro sys_exit 1
        push dword %1
        mov eax, SYS_EXIT
        call kernel
%endmacro

%macro sys_read 3
        push dword %3
        push dword %2
        push dword %1
        mov eax, SYS_READ
        call kernel
        add esp, 12
%endmacro

%macro sys_write 3
        push dword %3
        push dword %2
        push dword %1
        mov eax, SYS_WRITE
        call kernel
        add esp, 12
%endmacro
```

```
my_write_char:
        push eax
        mov esi, esp
        sys_write STDOUT, esi, 1
        add esp, 4
        ret

my_read_char:
        push dword 0
        mov edi, esp
        sys_read STDIN, edi, 1
        pop eax
        ret

global start
start:
        call my_read_char
        cmp eax, 0
        jz .exit_success
        jl .exit_failure
        call my_write_char
        jmp start
.exit_success:
        sys_exit EXIT_SUCCESS
.exit_failure:
        sys_exit EXIT_FAILURE
```

```
kernel:
        int 0x80
        ret

%define SYS_EXIT 1
%define SYS_READ 3
%define SYS_WRITE 4

%define STDIN 0
%define STDOUT 1
%define STDERR 2

%define EXIT_SUCCESS 0
%define EXIT_FAILURE 1

%macro sys_exit 1
        push dword %1
        mov eax, SYS_EXIT
        call kernel
%endmacro

%macro sys_read 3
        push dword %3
        push dword %2
        push dword %1
        mov eax, SYS_READ
        call kernel
        add esp, 12
%endmacro

%macro sys_write 3
        push dword %3
        push dword %2
        push dword %1
        mov eax, SYS_WRITE
        call kernel
        add esp, 12
%endmacro
```

```
kernel:
        int 0x80
        ret

%define SYS_EXIT 1
%define SYS_READ 3
%define SYS_WRITE 4

%define STDIN 0
%define STDOUT 1
%define STDERR 2

%define EXIT_SUCCESS 0
%define EXIT_FAILURE 1

%macro sys_exit 1
        push dword %1
        mov eax, SYS_EXIT
        call kernel
%endmacro

%macro sys_read 3
        push dword %3
        push dword %2
        push dword %1
        mov eax, SYS_READ
        call kernel
        add esp, 12
%endmacro

%macro sys_write 3
        push dword %3
        push dword %2
        push dword %1
        mov eax, SYS_WRITE
        call kernel
        add esp, 12
%endmacro
```

```
kernel:
%ifdef USE_TEST_KERNEL
        jmp test_kernel
%endif
        int 0x80
        ret
```

```
k




%
%
%

%
%define STDOUT 1
%define STDERR 2

%define EXIT_SUCCESS 0
%define EXIT_FAILURE 1

%macro sys_exit 1
        push dword %1
        mov eax, SYS_EXIT
        call kernel
%endmacro

%macro sys_read 3
        push dword %3
        push dword %2
        push dword %1
        mov eax, SYS_READ
        call kernel
        add esp, 12
%endmacro

%macro sys_write 3
        push dword %3
        push dword %2
        push dword %1
        mov eax, SYS_WRITE
        call kernel
        add esp, 12
%endmacro
```

```
kernel:
%ifdef USE_TEST_KERNEL
        jmp test_kernel
%endif
        int 0x80
        ret
```

```
%define STDERR 2

%define EXIT_SUCCESS 0
```

```
%define USE_TEST_KERNEL
%include "mylib.asm"

...

section .data
test_kernel_eax dd 0
test_kernel_stack_0 dd 0
test_kernel_stack_1 dd 0
test_kernel_stack_2 dd 0
test_kernel_one_time_jmp dd 0
section .text

test_kernel:
        cmp [test_kernel_one_time_jmp], dword 0
        je .call_normal_kernel
        jmp [test_kernel_one_time_jmp]
.call_normal_kernel:
        int 80h
        ret
```

```
kernel:
%ifdef USE_TEST_KERNEL
        jmp test_kernel
%endif
        int 0x80
        ret
```

```
%define STDERR 2

%define EXIT_SUCCESS 0
```

```
%define USE_TEST_KERNEL
%include "mylib.asm"

...

section .data
test_kernel_eax dd 0
test_kernel_stack_0 dd 0
test_kernel_stack_1 dd 0
test_kernel_stack_2 dd 0
test_kernel_one_time_jmp dd 0
section .text

test_kernel:
        cmp [test_kernel_one_time_jmp], dword 0
        je .call_normal_kernel
        jmp [test_kernel_one_time_jmp]
.call_normal_kernel:
        int 80h
        ret
```

```
check_flushing:
        mov [test_kernel_one_time_jmp], dword .my_fake_kernel
        call myio_flush
        mov eax, [test_kernel_eax]
        TEST_assert_eax_equals SYS_WRITE
        mov eax, [test_kernel_stack_0]
        TEST_assert_eax_equals dword STDOUT
        mov eax, [test_kernel_stack_1]
        TEST_assert_eax_equals myio_obuffer
        mov eax, [test_kernel_stack_2]
        TEST_assert_eax_equals 0
        ret
.my_fake_kernel:
        mov [test_kernel_one_time_jmp], dword 0
        mov [test_kernel_eax], eax
        mov eax, [esp+4]
        mov [test_kernel_stack_0], eax
        mov eax, [esp+8]
        mov [test_kernel_stack_1], eax
        mov eax, [esp+12]
        mov [test_kernel_stack_2], eax
        mov eax, [test_kernel_stack_2]
        ret
```

```
kernel:
%ifdef USE_TEST_KERNEL
        jmp test_kernel
%endif
        int 0x80
        ret
```

```
%define STDERR 2

%define EXIT_SUCCESS 0
```

```
%define USE_TEST_KERNEL
%include "mylib.asm"

...

section .data
test_kernel_eax dd 0
test_kernel_stack_0 dd 0
test_kernel_stack_1 dd 0
test_kernel_stack_2 dd 0
test_kernel_one_time_jmp dd 0
section .text

test_kernel:
        cmp [test_kernel_one_time_jmp],
        je .call_normal_kernel
        jmp [test_kernel_one_time_jmp]
.call_normal_kernel:
        int 80h
        ret
```

```
check_flushing:
        mov [test_kernel_one_time_jmp], dword .my_fake_kernel
        call myio_flush
        mov eax, [test_kernel_eax]
        TEST_assert_eax_equals SYS_WRITE
        mov eax, [test_kernel_stack_0]
        TEST_assert_eax_equals dword STDOUT
        mov eax, [test_kernel_stack_1]
        TEST_assert_eax_equals myio_obuffer
        mov eax, [test_kernel_stack_2]
        TEST_assert_eax_equals 0
        ret
.my_fake_kernel:
        mov [test_kernel_one_time_jmp], dword 0
        mov [test_kernel_eax], eax
        mov eax, [esp+4]
        mov [test_kernel_stack_0], eax
        mov eax, [esp+8]
        mov [test_kernel_stack_1], eax
        mov eax, [esp+12]
        mov [test_kernel_stack_2], eax
        mov eax, [test_kernel_stack_2]
        ret
```

```
global start
start:
        TEST_runtests check_empty_buffers
        TEST_runtests check_flushing
        TEST_runtests check_write_char
        TEST_runtests check_read_char
        TEST_runtests check_filling
        TEST_print_summary
        TEST_exit
```

Larrys buffered IO library

```
%include "mylib.inc"

section .data
myio_obuffer_len dd 0
myio_ibuffer_len dd 0
myio_ibuffer_idx dd 0

%define BUFFERSIZE 1024
section .bss
myio_obuffer resb BUFFERSIZE
myio_ibuffer resb BUFFERSIZE

section .text

; myio_flush
;   actually write whatever is in the output buffer to STDOUT
global myio_flush
myio_flush:
        pusha
        mov esi, myio_obuffer
        mov ecx, [myio_obuffer_len]
.try_again:
        sys_write STDOUT, esi, ecx
        cmp eax, 0
        jl .exit_with_failure
        add esi, eax
        sub ecx, eax
        jnz .try_again
        mov [myio_obuffer_len], dword 0
        popa
        ret
.exit_with_failure:
        sys_exit EXIT_FAILURE

; myio_write_char
;   put one character in the output buffer, flush if necessary
global myio_write_char
myio_write_char:
        pusha
        mov edi, myio_obuffer
        add edi, [myio_obuffer_len]
        mov [edi], al
        add [myio_obuffer_len], dword 1
        cmp [myio_obuffer_len], dword BUFFERSIZE
        jl .return
        call myio_flush
.return:
        popa
        ret
```

```
; myio_read_char
;   fetch next char (in eax) from input buffer,
;   fill if necessary
global myio_read_char
myio_read_char:
        push esi
        push edi
        push edx

        mov eax, 0
        mov edx, [myio_ibuffer_len]
        cmp edx, 0
        jg .fetch_next_char_from_buffer

        ; fill buffer
        mov [myio_ibuffer_idx], dword 0
        sys_read STDIN, myio_ibuffer, BUFFERSIZE
        mov edx, eax
        cmp eax, 0
        je .return

.fetch_next_char_from_buffer:
        mov esi, myio_ibuffer
        add esi, [myio_ibuffer_idx]
        mov eax, 0
        mov al, [esi]
        add [myio_ibuffer_idx], dword 1
        dec edx
        mov [myio_ibuffer_len], edx

.return:
        pop edx
        pop edi
        pop esi
        ret
```

!

Friday

Friday

Friday

seems like I don't need it before next month anyway

eat flaming death!