California State University, Sacramento
College of Engineering and Computer Science

Computer Science 35: Introduction to Computer Architecture

Spring 2022 – Lab 5 – *Gamblers*

Overview

Your work at Sutter's Mill has been both profitable and adventurous. As the days wind on, more and more able-bodied workers are coming to Coloma. The camp is growing in size, quite rapidly, with a wide assortment of people from all over the World.

When not sawing wood, there is considerable "down time". Some people use it to rest, some read, some hunt, and others spend it playing games of chance.

One of the new recruits brought three dice and introduced everyone to the game "Chuck-a-Luck". This a very, very basic game of chance. Three dice are thrown, and people bet on the outcome. There are a number of different types of bets, but the most common one is a "Triple" – all the dice are the same.

You have some extra time, let's play Chuck-a-Luck!

Your Task

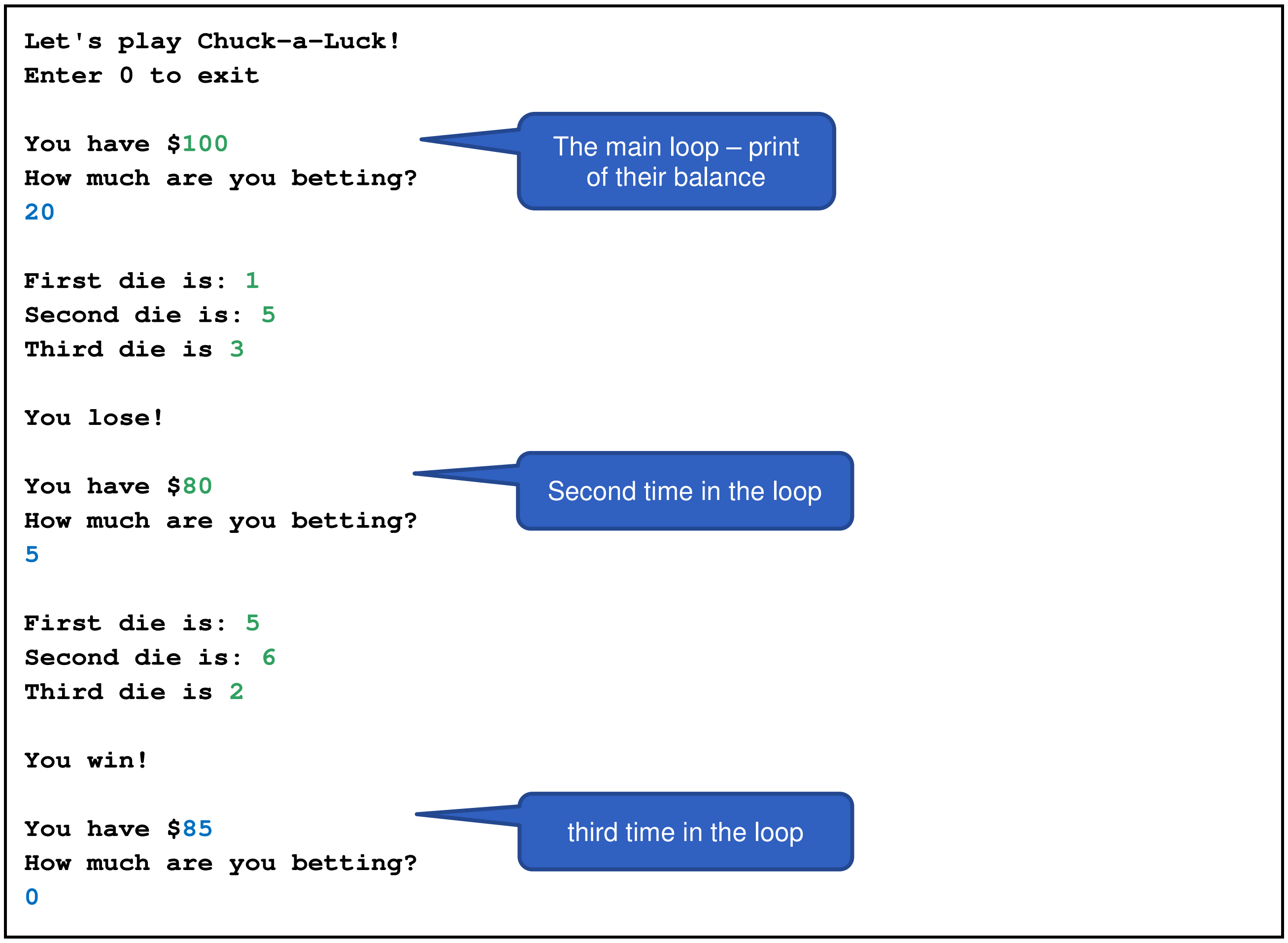
Players have the ability to bet one any number of possible outcomes. This can include if a specific number comes up, the sum of the dice, etc.... Your task to write a game that simulates Chuck-a-Luck – well, at least, the most basic version.

Your program will continue to loop while the user enters a bet > 0. You will then simulate the tossing of three dice. If any of the dice matches the user's selection, they win. Otherwise, they lose.

Just to make things a tad simpler, let's assume that **6** is the winning die number. So, if any of the dice is **6**, the player wins. The player will start the game with a credit of $100.

Example

The following is a sample run of the program. The user's input is printed in **blue**. The data outputted from your calculations is printed in **green**.



```
Let's play Chuck-a-Luck!
Enter 0 to exit

You have $100
How much are you betting?
20

First die is: 1
Second die is: 5
Third die is 3

You lose!

You have $80
How much are you betting?
5

First die is: 5
Second die is: 6
Third die is 2

You win!

You have $85
How much are you betting?
0
```

Tips

Work on each of the requirements below one at a time. You will turn in the final program, but incremental design is best for labs.

1. First, get the random numbers to work correctly. You should save all the values for later.

2. Get the If-Statement logic working.

3. Get the bet (and changing the player's total) working

4. Finally, only when #1 – #3 are done, work on the loop.

Necessary Instructions

Reading Integers

The CSC 35 Library has a subroutine called "ScanInt" that will read a value from the keyboard and store it into `rcx`. This is equivalent to the Java Scanner class method "nextInt".

`call ScanInt`	Java equivalent: `rcx = scanner.nextInt();`

Random Numbers

For the first part, you need to generate a random number. The csc35.o object library contains a subroutine called "random". Pass the range of numbers into **rcx**. It will return a random number from 0 to n-1 into **rcx**.

For example, if you store 100 into rcx and call the function, rcx will contain 0 to 99. So, how do you make the range 1 to 100? Perhaps you can add 1.

`mov rcx, 6`	rcx = range of numbers
`call Random`	rcx = a random number between 0 and 5
`add rcx, 1`	rcx now has a number between 1 to 6.

Requirements

1. Display an introductory message.
2. Loop until the player enters a bet ≤ 0
3. Display the player's balance.
4. Input the player's bet
5. Simulate rolling three dice. You might want to save these values for later in memory.
6. Tell the player if they win are lost.
7. Either add or subtract the bet from the total – depending on if they won/lost.

Program Pseudocode

```
loop
    output the player's balance
    input the player's bet

    if the bet ≤ 0 then leave the loop

    die1 = random number from 1 to 6
    die2 = random number from 1 to 6
    die3 = random number from 1 to 6

    output the die values

    if any die = 6 then
        output "You win!"
        add the bet to the total
    else
        output "You lose!"
        subtract the bet to the total
    end if

end loop
```

Submitting Your Lab

To submit your lab, you must run Alpine by typing the following and, then, enter your username and password.

```
alpine
```

To submit your lab, send the assembly file (do not send the a.out or the object file to:

```
dcook@csus.edu
```

⚠ **This activity may only be submitted in Intel Format.**

Using AT&T format will result in a zero. Any work from a prior semester will receive a zero.

Often, Chuck-a-Luck dice were put in a metal "birdcage" so they wouldn't get lost or be "modified" by unscrupulous players (i.e. cheaters).

The one that you have at Sutter's Mill, doesn't have a cage. (plot hint!)

UNIX Commands

Editing

Action	Command	Notes
Edit File	**`nano`** *filename*	"Nano" is an easy to use text editor.
E-Mail	**`alpine`**	"Alpine" is text-based e-mail application. You will e-mail your assignments it.
Assemble File	**`as -o`** *object source*	Don't mix up the *objectfile* and *asmfile* fields. It will destroy your program!
Link File	**`ld -o`** *exe object(s)*	Link and create an executable file from one (or more) object files

Folder Navigation

Action	Command	Description
Change current folder	**`cd`** *foldername*	"Changes Directory"
Go to parent folder	**`cd ..`**	Think of it as the "back button".
Show current folder	**`pwd`**	Gives the current a file path
List files	**`ls`**	Lists the files in current directory.

File Organization

Action	Command	Description
Create folder	**`mkdir`** *foldername*	Folders are called directories in UNIX.
Copy file	**`cp`** *oldfile newfile*	Make a copy of an existing file
Move file	**`mv`** *filename foldername*	Moves a file to a destination folder
Rename file	**`mv`** *oldname newname*	Note: same command as "move".
Delete file	**`rm`** *filename*	Remove (delete) a file. There is **no** undo.