California State University, Sacramento
College of Engineering and Computer Science

Computer Science 35: Introduction to Computer Architecture

Fall 2021 – Lab 4 – *Ravenclaw Tower*

Overview

Ah! You enjoy every moment of being a student at the famous ***Hogwarts School of Witchcraft and Wizardry***. Every student is sorted into one of the Four Houses with each having its own common room and, attached, dormitories.

The Hufflepuff Common Room is located close to the kitchens in is known for being warm and comfortable. The Slytherin Common Room is hidden in the dungeons and can only be entered by using a password. The Gryffindor Common Room is located high in the towers and, like the Slytherin's, requires a password.

The Ravenclaw Common Room is quite different.

Like Gryffindor, it is located in a tower. However, unlike Gryffindor, there is **no** password. Instead, the door challenges your ability to think.

Wit beyond measure is Man's greatest treasure.

This test can range from math problems to abstract riddles. Until you get the correct answer, you cannot enter. Often, outside the door, you can see several students sleeping – being far too tired to solve the challenge.

Your Task

Your task to write a game that will challenge the Ravenclaw student's ability to think. This game is also quite easy. The Door (computer) will select a random number from 1 to 100. Then the student will attempt to guess it. After each guess, the Door tells the student whether their guess is too high or too low. Once they get it correct, the swings open.

Basically, you are going to write a loop that will continue until the guess is equal to the correct answer. Inside the loop, you need to check if the correct answer is too high or too low and display a message. The exact wording is up to you. Make sure to print a third message when the loop is complete.

Part 1

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 **rax**. It will return a random number from 0 to n-1 into **rax**.

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

Part 2

This is the main game.

1. Loop until the user guesses the secret letter. The loop must check their guess against the correct answer.
2. If the user guesses too low or too high, print text telling the user.
3. When the game is over, print text congratulating the user.

Example

Your solution doesn't to look exactly like the example below. User input is displayed in **blue** (this is for readability, you don't have the make the input blue). For this example, the Door selected ***42*** as the secret number.

```
Greetings Ravenclaw student!
Yet... before... you may proceed... within 1 and 100... a number I need.

Guess: 50
Alas, that is too large.

Guess: 20
Regrettably, that is too small.

Guess: 40
Regrettably, that is too small.

Guess: 45
Alas, that is too large.

Guess: 42

Well-reasoned! You may now enter!
```

Requirements

1. Display an introductory message.
2. Create a random number
3. Loop until they enter the correct answer
4. Display a message if their guess is too high or too low.
5. Display a message when they guess correctly

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.

- First make sure your random number is working. You can print it to the screen for testing.
- Second, make your While Loop work. Don't worry about the If Statements.
- Finally, add the If Statements.

Submitting Your Lab

Please set the subject field of your e-mail to be:

CSc 35 – #

...where # is your lecture section number. This will help me sort your work.

To submit your lab, you must run Alpine by typing the following. You might have to re-enter your username and password.

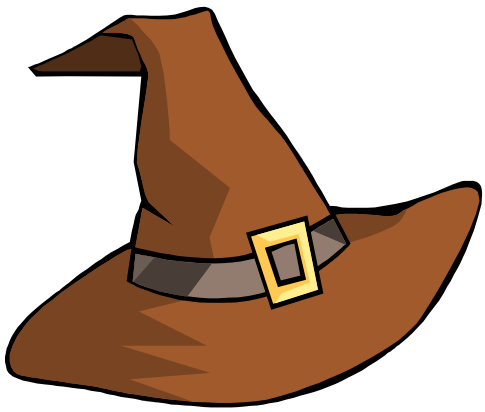
```
alpine
```

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

```
To      : dcook@csus.edu
```

Please give a descriptive subject for your e-mail. For example, the following is a good subject for a student in lecture Section 1.

```
Subject : CSC 35 – 1
```



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 *object* and *source* 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.