California State University, Sacramento
College of Engineering and Computer Science

Computer Science 35: Introduction to Computer Architecture

Spring 2021 – Lab 6 – *Magical Alphabet Soup*

Overview

Ah! You enjoy every moment of being a student at the famous ***Hogwarts School of Witchcraft and Wizardry***. This noble school has produced some of the finest magical students the World has ever seen.

At Hogwarts, everything is magical – from the paintings, to the walls, to even your nightly soup. The elves, in the Hogwarts kitchens, have prepared a wonderful version of the classic ***alphabet soup***. This soup tastes differently to everyone who eats it – being the flavor that enjoy the most.

Ah… magic!

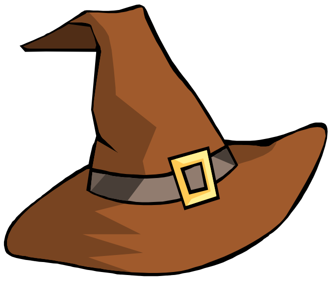
However, there is a catch. Before you can enjoy your dinner, you have to guess a secret letter.

Until you do, the soup simply cannot be eaten. There is an invisible barrier prevents you from eating a drop. The Weasley twins attempted to cast a counter-jinx, with disastrous results.

Guessing the Letter

This game is also quite easy. The soup (computer) will select a lowercase letter from ***a*** to ***z***. Then the student will attempt to guess it. After each guess, the soup tells the student whether their guess is too high or too low. Once they get the letter correct, the game is complete… and bon appetite!

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 want to generate the secret letter. Letters are just ASCII characters which, in turn, are just bytes – nothing more. You can compare two characters with each other just like any other number. But, you need to compare using 8-bit!

Read the documentation!

The csc35.o object library contains a subroutine called "random". Pass the range of numbers into %rbx. It will return a random number from 0 to n-1 into %rbx. Also look at an ASCII table to figure out the value of a lowercase ***a***. Then, see if you can generate a random number between ***a*** and ***z***.

You must figure this out on your own. I will not provide any help except to open the CSC35 library documentation. Make sure this part works before proceeding to Part 2.

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 soup selected ***g*** as the secret letter.

```
Hello! I am your delicious Hogwarts Alphabet Soup.
Before you can eat me, you must guess my letter!

Guess: c
Alas, you are too low.

Guess: m
Sorry, you are too high.

Guess: h
Sorry, you are too high.

Guess: f
Alas, you are too low.

Guess: g
Correct! You may now eat me!
```

Tips

- Use 8-bit registers. The library function that reads a single character is 8-bit
- You can only compare registers of equal size.
- 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.

Requirements

1. Display an introductory message. *(5 points)*
2. Create a random letter *(10 points)*
3. Loop until they enter the correct answer *(10 points)*
4. Display a message if their guess is too high or too low. *(10 points)*
5. Display a message when they guess correctly *(5 points)*

Submitting Your Lab

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.

Afterwards, run Alpine by typing the following and, then, enter your username and password.

```
alpine
```

Please send an e-mail to yourself (on your Outlook, Google account) to check if Alpine is working. To submit your lab, send the assembly file (not **`a.out`** or the object file) to:

```
dcook@csus.edu
```

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.