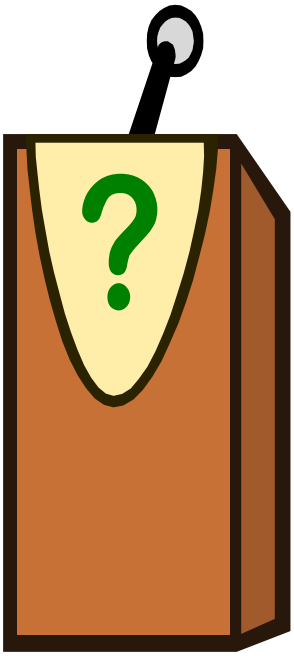
California State University, Sacramento
College of Engineering and Computer Science

Computer Science 35: Introduction to Computer Architecture

Spring 2017 – Project 1 – *Guess My Phrase*

Revision 3

Overview



It's time for Spring Break! Woo hoo! However, to keep your mind keen and sharp, you are going to write a game!.

We have all watched shows like "Wheel of Fortune". In this game, the player must guess a word or phrase only seeing part of it. Some letters are visible, and others are not. The first player will enter a phase (a word or words). The second player will then attempt to guess it. The challenge is that you will only display a few of the characters. Well, every second character. This will make it a tad challenging:

Gameplay

After the first player enters a phase, you will then implement a loop to replace every second character with a dash (or asterisk, question mark, etc...). It's your choice!

The game will then continue to prompt the second player until then guesses it correctly. For example, here is "**`Sacramento`**" displayed with different characters:

- **`S?c?a?e?t?`**
- **`S*c*a*e*t*`**
- **`S_c_a_e_t_`**

Pseudocode

The following pseudocode will give you a basic idea of what you have to do:

1. Display your program's name
2. Read the correct answer from the keyboard.
3. Implement a For Loop to make a copy of the correct answer
4. Implement a For Loop to replace every second character with something (asterisk, underscore, etc...)
5. Implement a While Loop
    a. Display the phrase (with missing characters)
    b. Input the player's guess
    c. Implement a For Loop to compare the correct answer to the guess.

Due Date

The assignment is due in 2 weeks after you return from Spring Break.

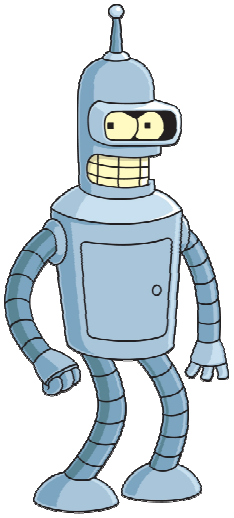
Sample Program

Here is a sample program. The player 1's input is printed in **red**. Player 2's input is displayed in **blue**.

```
Welcome to What's My Word!

Player 1, enter a word: Robot
```

Screen is cleared

```
Player 2, the word is R-b-t
Your Guess: Ribet
Sorry, that is wrong.

Player 2, the word is R-b-t
Your Guess: Rabit
Sorry, that is wrong.

Player 2, the word is R-b-t
Your Guess: Robot
Correct!
```



Tips

1. ***Use incremental design!*** Write your program in parts – get it to work a little bit more each time you assembly it. Never attempt to write the entire solution in just one take.... Take your time!

2. The UNIX/Linux read call returns the total number of bytes **read** from the keyboard. You can use this to control your for loops.

3. If the amount of characters the user entered for a guess is not equal to the number of characters in the correct answer – you can assume it isn't correct. Otherwise, you need to check each character.

4. I strongly recommend you create subroutines for reading and writing text. You might want to use push/pop to protect variables. It might even be wise to create subroutine for comparing two buffers.

Reading and Storing Text

To read text from the keyboard, please read about the ScanCString subroutine in the CSC35 Library. You will need to create a buffer large enough to hold the phrase.

The example below creates a buffer (space) of 30 characters.

```
Phrase:
    .space 30
```

Read the documentation on how to use ScanCString and LengthCString.

How to Connect from Home

Step 1 – Windows

The three servers that we use to do our labs cannot be accessed from off campus – at least directly. To connect these computers, first connect to Athena using Putty.

```
athena.csus.edu
```

Step 1 – Macintosh

Open the Terminal program. This is the same UNIX prompt that you get when you connect to Athena. Mac-OS X is a version of UNIX. Neat! Once at the prompt, type the following where ***username*** is your ECS username.

```
ssh username@athena.csus.edu
```

Step 2 – Secure Shell to SP1, SP2, or SP3

Once you are connected, you need to Secure Shell (SSH) to the SP computers. Basically, you will connect to Athena and it will connect you to the SP computer. Type the following at the UNIX prompt (this example uses SP2).

```
ssh sp2
```

You will have to enter your password again and (maybe) have to manually type "yes".

Requirements

YOU MUST DO YOUR OWN WORK. DO NOT ASK OTHER STUDENTS FOR HELP.

If you ask for help, both you and the student who helped you will receive a 0. Based on the severity, I might have to go to the University.

1. Display the title (of your choice) (5 points)
2. Read the incorrect phrases from the keyboard (20 points)
3. Clear the screen (10 points)
4. Loop until the user enters a correct answer (30 points)
5. Display the secret phrase (with the missing characters) each time in the loop (15 points)
6. When they are correct, display message congratulating them (10 points)
7. Proper formatting and comments (10 points)

Extra Credit

For an extra 20 points, keep track of how many time the user guesses. You can use comparisons to display a specialized message. For example, you could evaluate them like this:

Number of guesses	Letter Grade
`1 to 2`	A. Awesome work!
`3 to 4`	B. Best, that was!
`5 to 6`	C. Concentrate next time!
`6 to 7`	D. Don't quite your day job.
`8+`	F. Failure

This is just one possible way of doing it. You just need three different messages.

Submitting Your Project

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

```
alpine
```

To submit your lab, send the source 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** *objectfile asmfile*	Don't mix up the *objectfile* and *asmfile* fields. It will destroy your program!
Link File	**ld -o** *exefile objectfiles*	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 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.