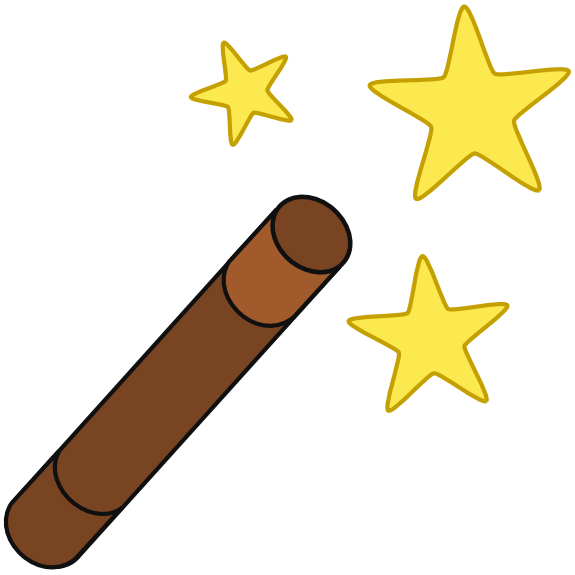
California State University, Sacramento
College of Engineering and Computer Science

Computer Science 35: Introduction to Computer Architecture

Spring 2018 – Lab 2 – *Twenty Points to Ravenclaw*

Overview

Student wizards and witches are taught 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.



During the school year – full of potions, spells, Quidditch, and various students being maimed – each House is awarded points by the faculty. At the end of the term, the House with the most points wins the prestigious House Cup!

These points can be earned by academics, winning contests, bravery, etc... The points can also be lost by breaking rules, poor academics, etc…

You are going to use the odd Muggle technology called "computers" to create a simple program to keep track of points. You will input the number of students that have earned/loss points in different categories. At the end, display how many points were earned.

Sample Run

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 **red**.

```
Getting an A        : 27 points
Cleaning your robes : 5 points
Being late to class : -11 points
Saying the V-Word   : -34 points

How many students got A's?
4
How many students cleaned their robes?
2
How many students were late to class?
3
How many students said the V-Word?
1
Ravenclaw gained 51 points!
```

Information text

Prompt the user

Calculated output

Requirements

You must think of a solution on your own. The requirements are as follows:

1. **Come up with 4 of your categories.** Don't use mine.
2. Display a table to the screen. *Please see above.*
3. Display a prompt, to the user, for each student count.
4. Input the number of students for each category.
5. Calculate the total number of points. Tip: use a register to create a running total.
6. Output the total number of points with some helpful text.

Hints

- Start off by getting the first multiplication to work and print the correct value.
- Now 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.

Hello World On x86 Linux

```
# lab1.s
# YOUR NAME HERE
#
# 1. Assemble : as -o lab1.o lab1.s
# 2. Link     : ld -o a.out lab1.o csc35.o
# 3. Execute  : a.out

.data                                   #Start the data section
Message:                                #Message is an address
    .ascii "Hello, world!\n\0"          #Create a buffer of ASCII

.text                                   #Start the text section
.global _start                          #Make the _start label public

_start:                                 #UNIX starts here
    mov  $Message, %rax                 #Put the address into rax
    call PrintCString                   #Execute the csc35.o subroutine

    call EndProgram                     #Execute the csc35.o subroutine
```

Submitting Your Lab

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 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.