**California State University, Sacramento**
**College of Engineering and Computer Science**

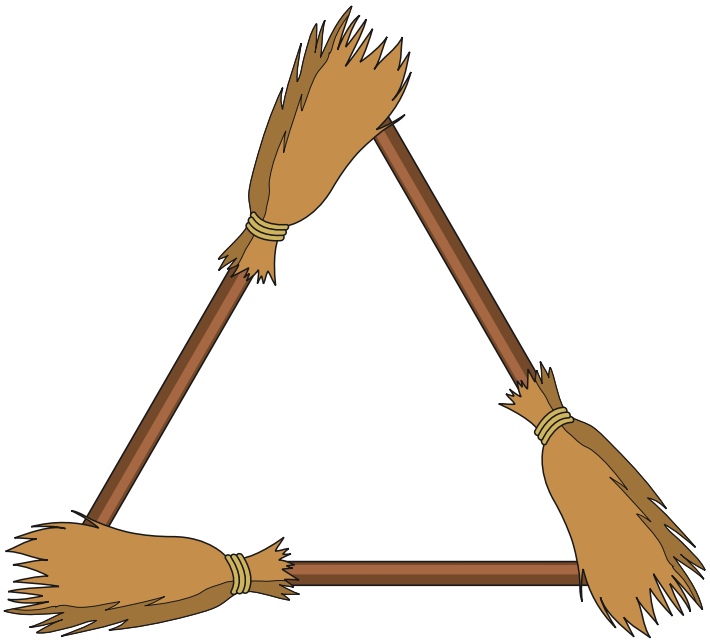
**Computer Science 35: Introduction to Computer Architecture**

**Spring 2021 – Lab 7 – *The Three Broomsticks***

## Overview

A short distance from ***Hogwarts School of Witchcraft and Wizardry*** sits a small town called Hogsmeade. This town is a common destination of students, faculty and tourists from all over the World. Hogsmeade contains a number of interesting shops including:

- *Honeydukes*, which offers a wide variety of magical candy including Every Flavor Beans, Fizzing Whizzbees, and Drooble's Best Blowing Gum.
- *Hog's Head Inn*, a dingy bar frequented by ne'er-do-wells.
- *Zonko's Joke Shop*, which sells "cheeky" magical items such as Dungbombs, Frog Spawn Soap, and Nose-Biting Teacups.
- *Madam Puddifoot's Tea Shop*, a popular destination for romantic evenings.



But, by far, the most popular destination for food and conversation is ***The Three Broomsticks***. As a popular meeting place, patrons tend to create a rather sizeable bill. So, just like muggles, Witches and Wizards have to split the bill.

You are going to use the odd muggle technology called "computers" to split the bill. Your program will read the costs and order name from a table.

## Sample Run

The user's input is printed in **blue**. The data outputted from your calculations is printed in **red**.

```
Welcome to The Three Broomsticks. Here is today's menu:

1. Bertie Bott's Every Flavor Sliders (823 knuts)
2. Mobius strip Pancakes (677 knuts)
3. Cauldron Cakes (1937 knuts)
4. Cornish Pasties (759 knuts)
5. Pumpkin Pasties (1237 knuts)

What is your order?
1

Your party enjoyed:
1. Bertie Bott's Every Flavor Sliders (823 knuts)

How many people are splitting the bill?
3

Okay, witches and wizards, give 274 knuts each.
```

A simple divide. …or is it?

## Tips for the Data Section

Here a few tips on how to structure your program.

- Create strings for each item in your vending machine.
- Create a table of those addresses (to lookup the purchased item).
- Also create a table of costs.

The following contains an example of how to make a table of addresses (which are quads) and table of values (also quads). Please feel free to change your labels.

```
Sliders:
   .ascii "1. Bertie Bott's Every Flavor Sliders (823 knuts)\n\0"

Pancakes:
   .ascii "2. Mobius strip Pancakes (677 knuts)\n\0"

   …

Items:
    .quad Sliders
    .quad Pancakes
    …

Costs:
   .quad 823
   .quad 677
   …
```

## Tips for the Text Section

- **Work in your program in parts – incremental design!**
- The user is entering a 1-indexed number. This means the first item (for the customer) is 1. However, this is the index 0 in the table. **Subtract 1**.
- Remember to use the correct scale for quads!
- Since the Items table contains address, you should use MOV rather than LEA. You want the address, not the address of the address.

## Input Validation

There are two problems that can occur in your program. Both of these require input validation. This programming technique prevents errors from occurring by making sure data is "valid" before continuing.

### *Invalid Index*

Since you are using table, you need to check if the index is valid. So, you need to create an **input validation loop**. Your program will not proceed until the user enters a valid value.

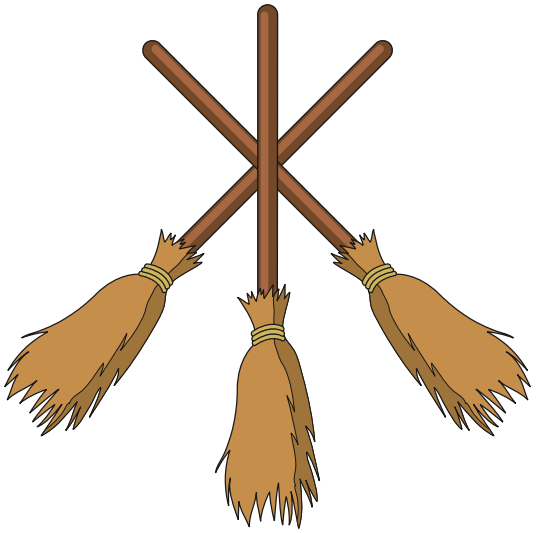
```
…
What is your order?
-1

What is your order?
100

What is your order?
0

What is your order?
5

Your party enjoyed:
5. Pumpkin Pasties (1237 knuts)
```



***Divide by Zero***

What happens in the user enters zero for the number of people splitting the bill? Yes, you divide by zero!

While this might not be an issue in the Magical World, it causes a great deal of problems for muggles! So, you need to create an **input validation loop**. Your program will not proceed until the user enters a positive integer.

```
…

Your party enjoyed:
2. Mobius strip Pancakes (677 knuts)

How many people are splitting the bill?
0

How many people are splitting the bill?
-1

How many people are splitting the bill?
5

Okay, witches and wizards, give 135 knuts each.
```

A simple divide. …or is it?

## Requirements

You must think of a solution on your own. **You can come up with your own theme**. You don't have to use mine. The requirements are as follows:

1. Display a menu of items and costs. You must have (at least) five *(5 points)*
2. Input their selection *(5 points)*
3. Output the item they bought to the screen. You must use a table. *(5 points)*
4. Input the number of guests *(5 points)*
5. Use input validation to prevent an invalid table index *(5 points)*
6. Use input validation to division by zero. *(5 points)*
7. Calculate and output the correct split. *(10 points)*

## Submitting Your Lab

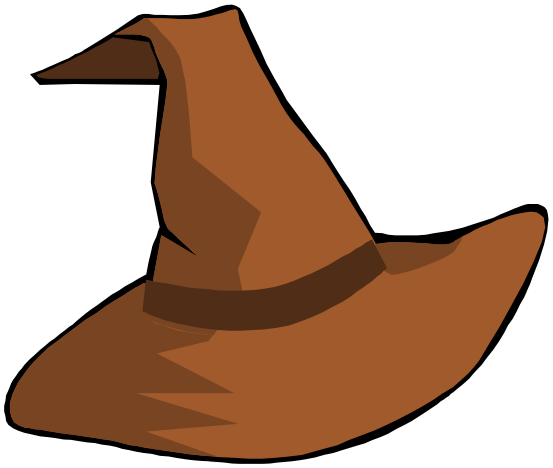
**This activity may only be submitted in Intel Format. Using AT&T format will result in a zero.**

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. |