California State University, Sacramento
College of Engineering and Computer Science

Computer Science 35: Introduction to Computer Architecture

Fall 2020 – Lab 6 – *Magical Vending Machine!*

Overview

Whether you are working a nine-to-five job or running from class to class, there is one machine that is in all our lives – the vending machine.

The student common rooms at ***Hogwarts School of Witchcraft and Wizardry*** are no different.

Sitting majestically, a vending machine graces each the four common rooms: Ravenclaw, Gryffindor, Hufflepuff and Slytherin. These wonderful devices contain a myriad of magical items that students need (since the school is quite accident prone).

You are going to use the odd muggle technology called "computers" to simulate a vending machine.

Your program will display a menu of items, input the amount of knuts, input their choice and, then, output their selection and their change.

Example

Your solution doesn't have to look exactly like the example below. The user's input is printed in **blue**. The data outputted from your calculations is printed in **red**. You don't have to make the text that color in your program.

```
Ravenclaw Common Room Vending machine

1. Quill & Ink (25 knuts)
2. One-Day-Only Cauldron (85 knuts)
3. One-Day-Only Wand (120 knuts)
4. Every Flavor Beans (42 knuts)

How many knuts were entered?
200

Enter your selection:
3

You selected:
3. One-Day-Only Wand (120 knuts)

Your change is 80 knuts.
```

Input Validation: Enough Money

But what happens if the student attempts to buy an item with insufficient funds? Well, you can't let them rob the vending machine! Right? You don't want them to be sent to Azkaban!

So, when they select and item, make sure they have enough. Naturally, you need a nice Loop to accomplish this task.

```
Ravenclaw Common Room Vending machine

1. Quill & Ink (25 knuts)
2. One-Day-Only Cauldron (85 knuts)
3. One-Day-Only Wand (120 knuts)
4. Every Flavor Beans (42 knuts)
5. Cancel the order (0 knuts)

How many knuts were entered?
50

Enter your selection:
3

Enter your selection:
4

You selected:
4. Every Flavor Beans (42 knuts)

Your change is 8 knuts.
```

Not enough money for a One-day-only Wand

Input Validation: Valid Selection

Since you are using an index into a table, you need to make sure that the index is, in fact, **valid**. So, add control logic that prevents the student from entering an invalid selection.

```
Enter your selection:
-1

Enter your selection:
12

Enter your selection:
4

You selected:
4. Every Flavor Beans (42 knuts)
```

-1 and 12 are not valid entries

Have fun!

You can come up with your own theme. You don't have to use mine. Use your imagination!

- Cat items
- Dog items
- Rick and Morty items
- Pokemon items
- etc....

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.

```
Quill:
   .ascii "1. Quill & Ink (25 knuts)\n\0"

Cauldron:
   .ascii "2. One-Day-Only Cauldron (85 knuts)\n\0"

   …

Items:
    .quad Quill
    .quad Cauldron
    …

Costs:
   .quad 25
   .quad 85
   …
```

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 student) 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.

Requirements

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

1. Display a menu of items and costs. You must have (at least) five. The last one should cost zero – which will let the user get back their money. Make sure to also print a name for your vending machine.
2. Input how much money they entered
3. Input their selection
4. Output the item they bought to the screen. You must use a table.
5. Output their change to the screen. You must use a table to look up the cost.
6. Use input validation to prevent an invalid table index
7. Use input validation to prevent the user from buying something they can't afford.

Extra Credit: 10 points

Vending machines like to dispense coins in (at least) two different denominations. For 10 extra points compute and output the number of quarters and pennies (or knuts and sickles). Hint: use IDIV.

```
Ravenclaw Common Room Vending machine

1. Quill & Ink (25 knuts)
2. One-Day-Only Cauldron (85 knuts)
3. One-Day-Only Wand (120 knuts)
4. Every Flavor Beans (42 knuts)

How many knuts were entered?
200

Enter your selection:
3

You selected:
3. One-Day-Only Wand (120 knuts)

Your change is 80 knuts.
Dispensing 4 sickles (s) and 12 knuts(s)
```

1 sickle = 17 knuts.

So, 80 was divided by 17

Submitting Your Lab

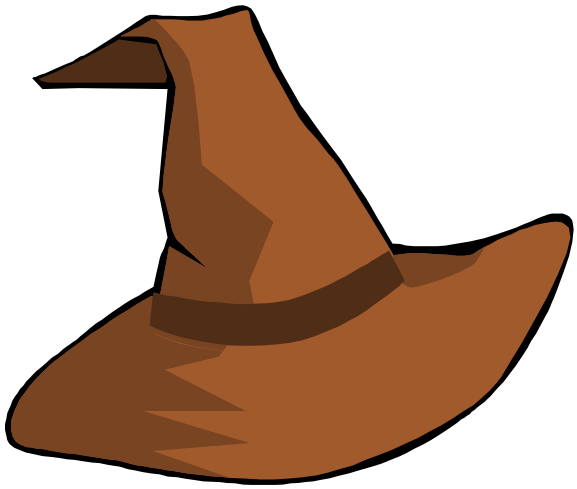
This activity may only be submitted in Intel Format.
Using AT&T format will result in a zero. Any work from a prior semester will receive a zero.

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.