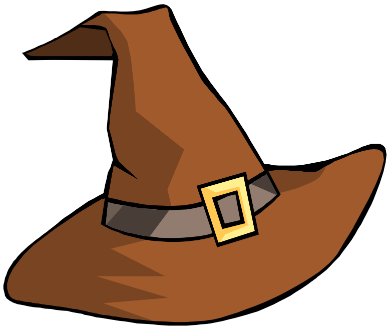
California State University, Sacramento
College of Engineering and Computer Science

Computer Science 35: Introduction to Computer Architecture

Fall 2017 – Lab 3 – *Hogwarts*

Overview



Ah, to be a wizard! The ability to cast spells to clean your room, create gas for your car... or just make it fly. In the Harry Potter books, each student at Hogwarts is sorted into one of the four different houses. This is accomplished with an ancient hat called the Sorting Hat.

You might belong in Gryffindor,
Where dwell the brave at heart,
Their daring, nerve, and chivalry
Set Gryffindors apart;

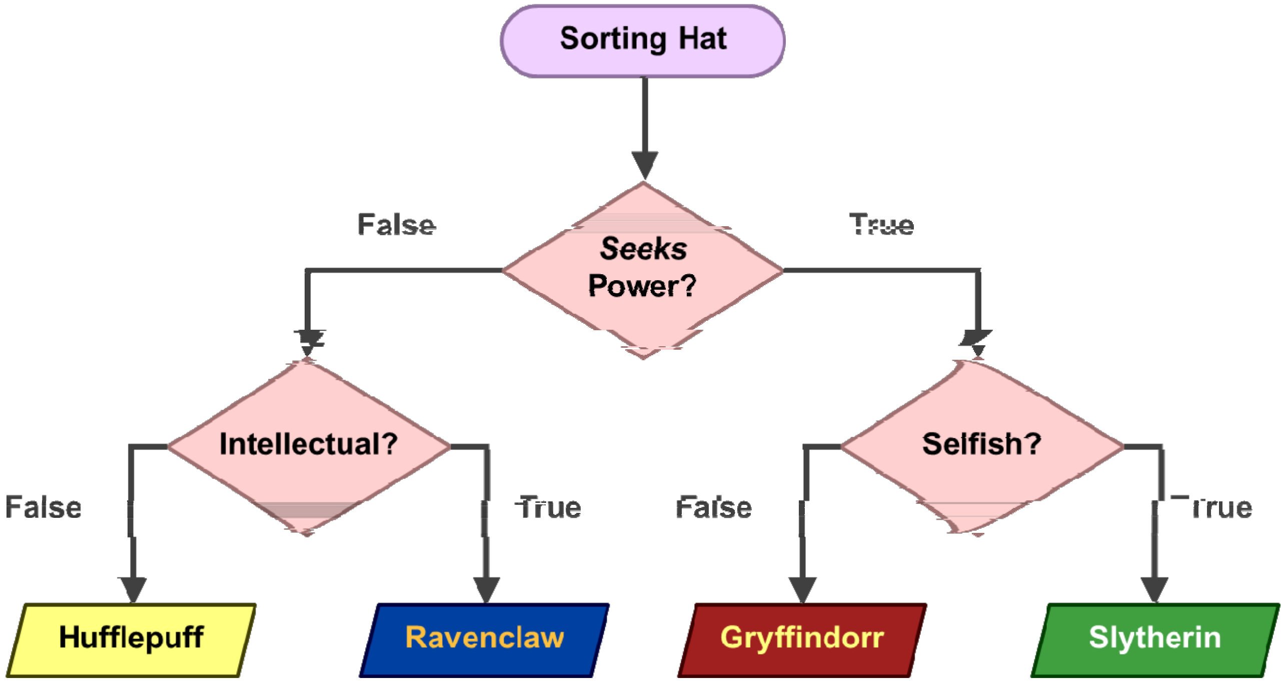
You might belong in Hufflepuff,
Where they are just and loyal,
Those patient Hufflepuffs are true
And unafraid of toil;

Or yet in wise old Ravenclaw,
if you've a ready mind,
Where those of wit and learning,
Will always find their kind;

Or perhaps in Slytherin
You'll make your real friends,
Those cunning folks use any means
To achieve their ends.

The Sorting Hat's Algorithm

Even though the Sorting Hat is revered, its job is not that hard. In fact, the Sorting Hat really doesn't do that much work. It simply senses the answers to a few questions and then puts the student into the correct house (hopefully). The logic is as follows:



Your Task

So, for your assignment, you get to create the program to recreate the Sorting Hat. You will ask questions and the user will respond with a yes or no. Based on their answers, you will print off the house they will put into.

Have Fun!

Don't use the four houses from Harry Potter. It might be fun to create your own four categories. Naturally, the questions the will be different, but the overall approach is the same.

The following are some example categories.

- What class in a role player game you are
- What cartoon character are you?
- Which Rick and Morty alien are you?
- What type pet should you get – species, breed, etc...
- What type of geek are you
- What political philosophy you are
- What music you should listen too
- etc...

Requirements

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

1. Display text telling the user what they are being sorted into (Hogwarts, party, etc…)

2. Input each choice (with prompts)

3. Implement the logic in the flowchart above. Use nested ifs

4. Display the output for all four possibilities.

Tips

- Like all labs, **build it in pieces**. First get a single If-Statement to work. Then, you can work on the more detailed ones.

- All labels **must** be unique. Choose your names well.

Examples

This is the output from one possible solution. Your solution doesn't to look exactly like the example below. But, make sure to fulfill all the requirements. Input is displayed underlined.

```
Welcome to Hogwarts! You will be sorted into one of the four noble houses. The
Sorting Hat is being placed upon your head.

Do you want power? (1=yes, 2=no)
2
Are you intellectual? (1=yes, 2=no)
1
Ravenclaw!
```

```
Welcome to Hogwarts! You will be sorted into one of the four noble houses. The
Sorting Hat is being placed upon your head.

Do you want power? (1=yes, 2=no)
1
Do want it for yourself? (1=yes, 2=no)
2
Gryffindor!
```

```
Welcome to Hogwarts! You will be sorted into one of the four noble houses. The
Sorting Hat is being placed upon your head.

Do you want power? (1=yes, 2=no)
1
Do want it for yourself? (1=yes, 2=no)
1
Slytherin!
```

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.