## Overview

Ah, to be a wizard!

The ability to cast spells to clean your room, create gas for your car... or to 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;

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

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

Or perhaps in Slytherin
You'll make your real friends,
Those cunning folks use any means
To achieve their ends.
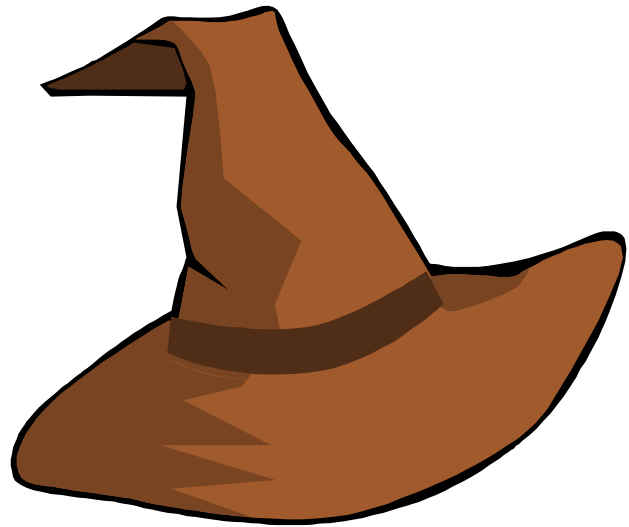
So, how does this magical relic work? What logic does it use to sort each student?
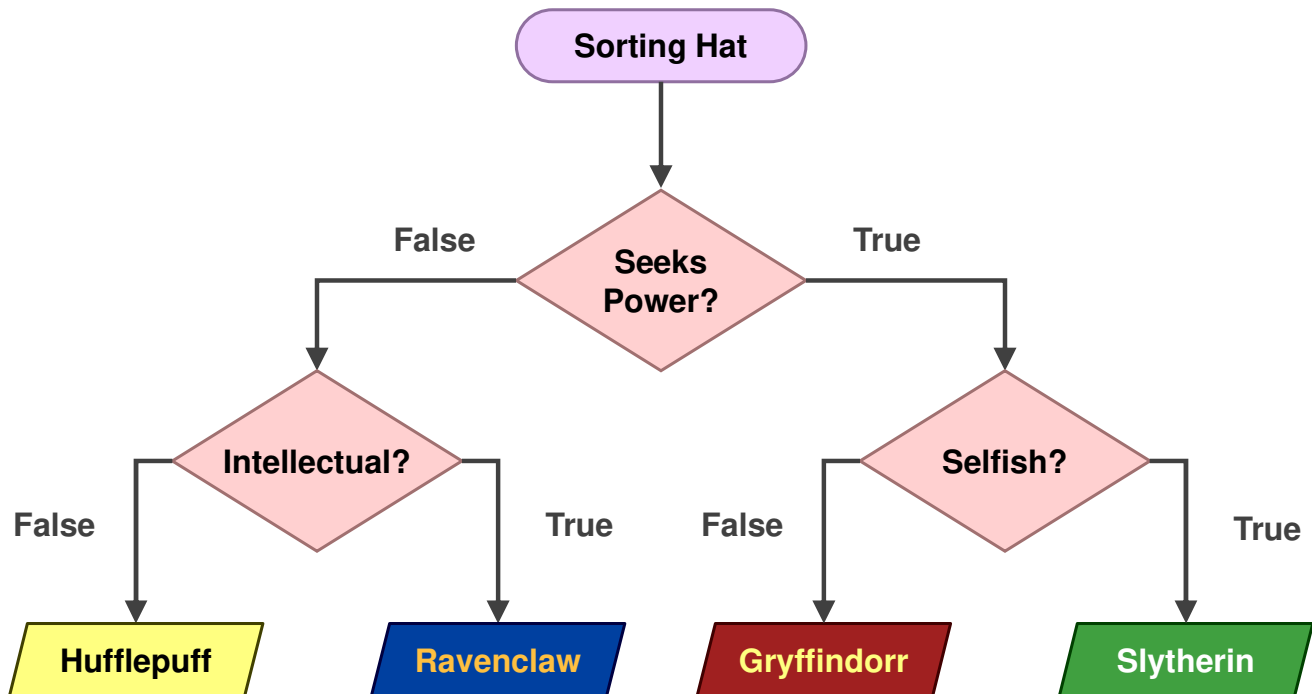
Your Task

So, for your assignment, you get to create a program to recreate the Sorting Hat. You will ask questions and the user will respond with a numeric value. Based on their answers, you will print which house they will be put into.

The Sorting Hat's Algorithm

While the Sorting Hat seems mysterious and powerful, it follows a basic nested If Statement. Well, this isn't really true, but it's your instructor's attempt to turn it into a programming assignment.

So, for this assignment, let's **assume** that it simply senses the answers to a few questions and then puts the student into the correct house (hopefully). The logic is as follows:

```
                        Sorting Hat
                             |
                             v
        False           Seeks              True
       /                Power?                 \
      v                                         v
  Intellectual?                              Selfish?
 /            \                             /         \
False         True                      False          True
v             v                          v              v
Hufflepuff   Ravenclaw              Gryffindorr      Slytherin
```

Have Fun!

You don't have to use the four houses from Harry Potter. Create your own four categories. Naturally, the questions 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
- What should you have for dinner
- What meme are you most resemble
- Which Rick and Morty alien are you
- What type pet should you get – species, breed, etc...
- What political philosophy you are
- What music you should listen too
- etc...

Examples

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

```
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 wish to (1) command/control others or (2) work behind the scenes?
2
Do you find contentment in (1) learning or (2) expression?
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 wish to (1) command/control others or (2) work behind the scenes?
1
Is it more important to help (1) yourself or (2) others?
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 wish to (1) command/control others or (2) work behind the scenes?
1
Is it more important to help (1) yourself or (2) others?
1
Slytherin!
```

Tips

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

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

- Assembly doesn't have blocks… so don't think in those terms. Your program will be structured far differently from programs in Java.

Requirements

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

1   You don't have to use Direct Storage. This program simply uses the value put into rbx by ScanInt.

2. Create your own Sorting Hat. Don't create the Hogwarts one that I used. Use your imagination.

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

4. Input each choice (with prompts). You <u>must</u> input a numeric value. Any lab that uses characters will receive a zero.

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

6. Display the output for all four possibilities.

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 (<u>not</u> `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.