



California State University, Sacramento
College of Engineering and Computer Science

Computer Science 35: Introduction to Computer Architecture

Spring 2023 – Project – *Wizard Battle (for grades)*

Overview

Your first meal at **Hogwarts School of Witchcraft and Wizardry** is going incredibly well.

So far, you have gorged yourself on a nearly endless supply of fried chicken, beef stroganoff, miso soup, garlic bread, mashed potatoes, roasted vegetables, and paneer makhani. And, that was just the first course!

After eating your desserts, you feel fit to burst. Fortunately, for you, you were able to stay under the ridiculous sugar limit set by the Department for the Enthusiastic Promotion of Student Health and the Ridiculously Overzealous Regulation of Sugary Intake. It wasn't hard, in fact. You already felt like a stuffed turkey before dessert.

Professor McGonagall stands and addresses the room. She waits a few moments for the slurping and munching sounds to trail away. "Well, I'm glad your stomachs are full; yet your minds are still quite empty. Nevertheless, before I let you return to your common rooms, we must first sing the Hogwarts School Song."

There is a school song? You don't know it? How can sing a song you have never heard? You are about to ask the Prefect, Joe Gunchy, about it, but are stopped short when McGonagall clears her throat. With a graceful flick of her wand, gleaming red words appear in the air and the students begin to sing... albeit poorly.



*Hogwarts, Hogwarts,
Hoggy Warty Hogwarts,
Teach us something, please,
Whether we be old and bald
Or young with scabby knees,
Our heads could do with filling
With some interesting stuff,
For now they're bare and full of air,
Dead flies and bits of fluff,
So teach us things worth knowing,
Bring back what we've forgot,
Just do your best, we'll do the rest,
And learn until our brains all rot.*



As the students sing, Professor McGonagall conducts using her wand, as a baton, in a desperate attempt to keep some semblance of order. But, it isn't working. Students are singing at different times and with increasingly awkward notes. Professor McGonagall occasionally winches when a student bellows a note reminiscent of fingernails on a chalkboard.

When the song ends, with some students finishing before others, McGonagall smiles broadly. You wonder if she is smiling *because* the song ended.

"And now, students, time for bed. Your classes start tomorrow, and I expect all of you to be well rested."

The Game

Overview

Like the classic Oregon Trail (created in 1971), your game is going to create a simple simulation based on resource management. In this style of game, you attempt to maximize your score by using your resources and obtaining assets. In your case, you will be creating a (very simple) game based on the life of a Hogwarts Student.

The game takes place over a 120-day period (about 4 months). You will start with zero knowledge, zero stress, and full (100%) endurance. So, basically, you start off the game as a new, happy, Hogwarts student that is hungry for knowledge. However, each week you will lose some of your knowledge and endurance. In addition, your stress level will steadily increase (making it quite hard to study). If your endurance gets to zero, the game ends automatically.



All your decisions will have an impact on the game – and whether you win or lose. Your goal is to survive the semester with the most knowledge possible. Each week can be broken into two logical sections:

Part 1: Saturday

When you reach the weekend (i.e. Saturday), you have to decide how to spend your time off. You can

- Rest – which will increase your endurance between 30% to 60%.
- Study – which will increase your knowledge between 15% to 30%. However, if your stress is over 100%, you cannot concentrate and won't learn anything!
- Play Quidditch – which will decrease your stress between 30% to 60%.

Part 2: A Week at Hogwarts

- Your endurance drops between 10% and 20% each week. Carrying those books and normal classroom injuries are really going to take a toll. If your endurance reaches 0%, the game ends automatically (and you lose).
- You will forget somewhere between 1% and 5% of your knowledge each week. So, you have to keep studying!
- Your stress increases between 10% and 30% each week. If it reaches 100%, you cannot study.

Sample Output

Your solution doesn't have to look exactly like this. However, this should show you the basic gameplay. For readability, the user's input is displayed in **red** and computer values are in **green**. You don't have to use color (unless you are going for extra credit). *As always, please feel free to change the wording of the text.*

```
=====
WIZARD BATTLE FOR GRADES
=====

Rules:
1. 120 days (1 semester)
2. Your resources
  * Your endurance drops 10-20% each week. If it reaches 0%, the game ends.
  * You will forget 1-5% of your knowledge each week.
  * Your stress increases 10-30% each turn. If it reaches 100%, you cannot study.
3. Your choices:
  * Resting increases endurance (30-60%).
  * Studying increases knowledge (15-30%).
  * Playing decreases stress (30-60%).

The goal is to survive the semester with the most knowledge possible.

120 DAYS LEFT!

Your endurance is at 100%
Your knowledge is at 0%
Your stress is at 0%
It's Saturday! Do you want to 1. Rest, 2. Study, 3. Play quidditch
2

You gained 13% knowledge from studying.

You lost 20% endurance.
You forgot 2% of what you studied.
You gained 18% stress.

113 DAYS LEFT!

Your endurance is at 80%
Your knowledge is at 11%
Your stress is at 18%
It's Saturday! Do you want to 1. Rest, 2. Study, 3. Play quidditch
2

You gained 20% knowledge from studying.

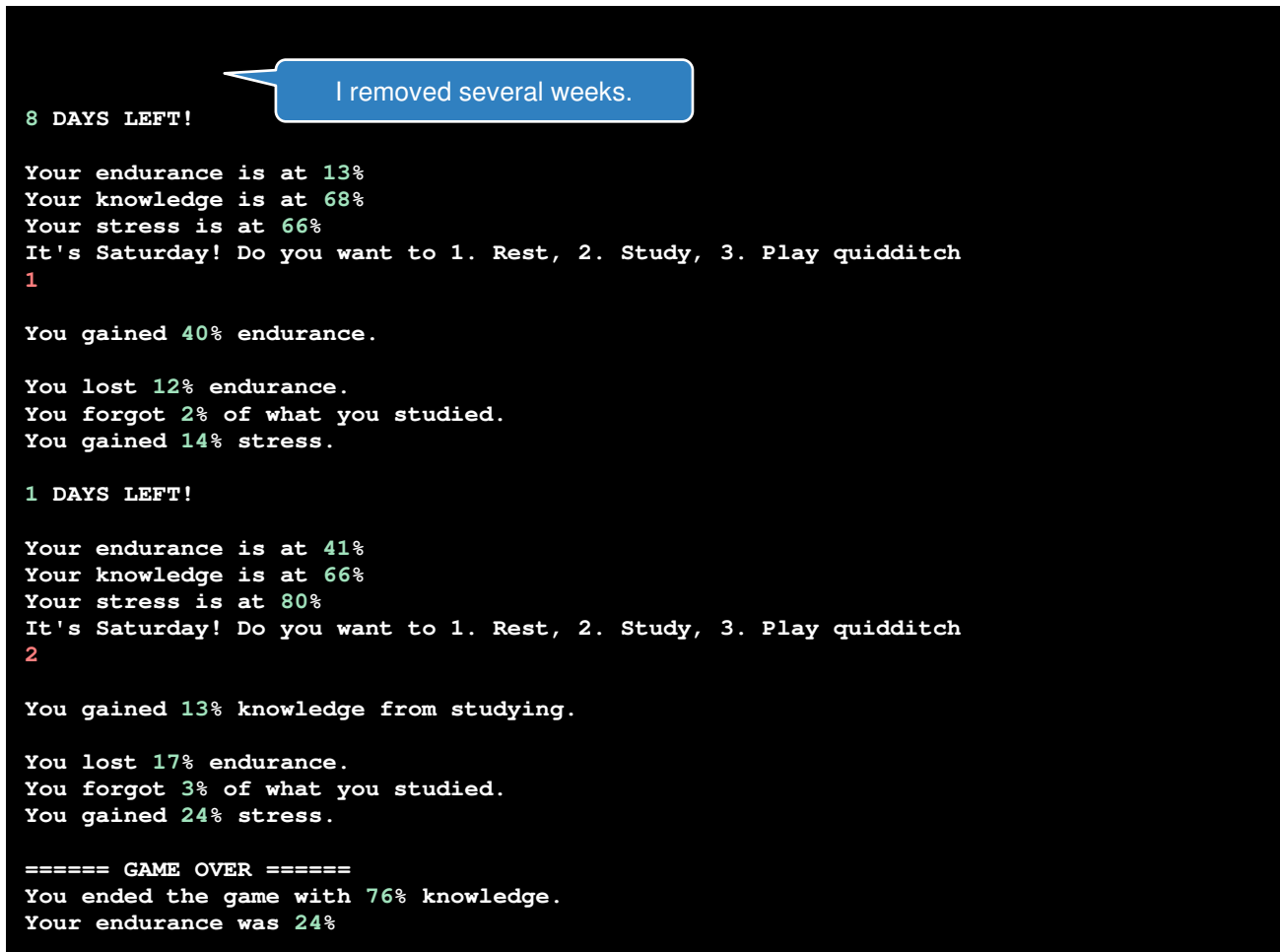
You lost 16% endurance.
You forgot 2% of what you studied.
You gained 28% stress.

106 DAYS LEFT!

Your endurance is at 64%
Your knowledge is at 29%
Your stress is at 46%
It's Saturday! Do you want to 1. Rest, 2. Study, 3. Play quidditch
3

You lost 40% stress by doing something fun.

You lost 17% endurance.
You forgot 1% of what you studied.
You gained 12% stress.
```



```

8 DAYS LEFT!

Your endurance is at 13%
Your knowledge is at 68%
Your stress is at 66%
It's Saturday! Do you want to 1. Rest, 2. Study, 3. Play quidditch
1

You gained 40% endurance.
You lost 12% endurance.
You forgot 2% of what you studied.
You gained 14% stress.

1 DAYS LEFT!

Your endurance is at 41%
Your knowledge is at 66%
Your stress is at 80%
It's Saturday! Do you want to 1. Rest, 2. Study, 3. Play quidditch
2

You gained 13% knowledge from studying.
You lost 17% endurance.
You forgot 3% of what you studied.
You gained 24% stress.

===== GAME OVER =====
You ended the game with 76% knowledge.
Your endurance was 24%

```

Getting the Demo Program

You can download a copy of the game (in its most basic form). **Type the following verbatim:**

```
curl devincook.com/csc/35/wizard.out > wizard.out
```

Once you have downloaded the file, you need to change its access rights so you can execute it. Type the following:

```
chmod 500 wizard.out
```

You'll learn more about this command in CSC 60. Now, to run the program type the following:

```
./wizard.out
```

Getting the Library

If you accidentally damaged or deleted the CSC 35 Library File, **type the following verbatim:**

```
curl devincook.com/csc/35/csc35.o > csc35.o
```

Random Numbers

The library has a built-in subroutine called "Random" that you must use to make your project work.

You will pass the **range** of numbers into **rdi**. After you call Random, rdi will contain a random number between 0 to n - 1 into **rdi**. For example, if you store 100 into **rdi** and call the subroutine, **rdi** will contain 0 to 99.

So, for example, do you create a random number between 10 and 30? Well, let's look at the actual range (or, in other words, how many numbers there are). Here, the range is 21 which is computed using $(30 - 10 + 1)$. If we call Random with **rdi** set to 21, it will return a number between 0 and 20. So, how do we make it a value between 10 and 30? Just add 10!

In general, this is the mathematical equation that is used for getting a random number with a *minimum* value and a given *range*:

```
minimum + Random(range)
```

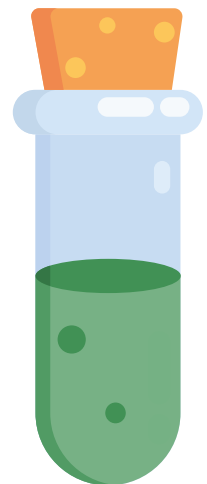
The actual assembly, for the example above, will look like the following:

```
mov    rdi, 21
call   Random          #rdi = 0 .. 20
add    rdi, 10          #rdi = 10 .. 30
```

Have Fun

Use your imagination. Your game doesn't have to be the Gold Rush. You can base your game on any fun theme that you want. But, only if you keep the same gameplay. For example, here are some possible scenarios:

- Kittens
- Cartoon: Spongebob Squarepants, Rick and Morty, Archer, etc....
- Politics
- Movies: comedy, sci-fi, horror, etc...
- Video games
- Television programs
- Characters from a book
- etc...



Project Pseudocode

```

assign Endurance = 100, Knowledge = 0, Stress = 0, Days = 120

while (Days > 0 and Endurance > 0)
    ... Summary
    output Days, " DAYS LEFT!"
    output "Your Endurance is at ", Endurance, "%"
    output "Your knowledge is at ", Knowledge, "%"
    output "Your Stress is at ", Stress, "%"

    ... Part 1: Saturday
    output "It's Saturday! Do you want to 1. Rest, 2. Study, 3. Play quidditch"
    input Choice

    switch (Choice)
        case 1:
            assign Value = random(30, 60)
            add Value to Endurance
            output "You gained ", Value, "% endurance."

        case 2:
            if (Stress < 100)
                assign Value = random(10, 20)
                add Value to Knowledge
                output "You gained ", Value, "% knowledge from studying."
            else
                output "You are too stressed to study!"
            end if

        case 3:
            assign Value = random(30, 60)
            subtract Value from Stress
            output "You lost ", Value, "% stress by doing something fun."
    end switch

    ... Part 2: Week at Hogwarts
    ... Endurance
    assign Value = random(10, 20)
    subtract Value from Endurance
    output "You lost ", Value, "% Endurance."

    ... Lost Knowledge
    assign Value = random(1, 5)
    subtract Value e from Knowledge
    output "You forgot ", Value, "% of what you studied."

    ... Gained Stress
    assign Value = random(10, 30)
    add Value to Stress
    output "You gained ", Value, "% stress."

    ... Remove a week
    subtract 7 from Days
end while

```

Tips

- **DON'T** attempt to write the entire program at one time. If you do, you won't be able to debug it. Experienced programmers use incremental design. Make a basic program and, very slowly, add the features you need.
- So, first get the main loop working... then, bit by bit, add the rest of the functionality.
- If you get stuck in an infinite loop, you can press Control+C to exit any UNIX program.

Requirements



YOU MUST DO YOUR OWN WORK. DO NOT ASK OTHER STUDENTS FOR HELP.

If you ask for help, both you and the student who helped you will receive a 0. Based on the severity, I might have to go to the University.



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

1. **Print the title of your program.** (5 points)
2. **Print the game rules. Let the player know how the game works!** (5 points)
3. **Loop for 120 days.** (10 points)
If you change the project theme, please feel free to change this value (e.g. 12 for months)
4. **Exit if there endurance drops below zero.** (10 points)
If you do the extra credit below, the game won't exit, but something else will happen.
5. **Part 1: Decision.** (20 points)
Input the player's choice. There needs to be at least 3 choices. The program must do different things based on the input.
6. **Unable to study.** (20 points)
If your stress is over 100%, you cannot gain any knowledge from studying. Display an error message if they are too stressed out.
7. **Part 2: Lost knowledge, lost endurance, and gained stress.** (15 points)
Your program must calculate how much they knowledge they lost, how much endurance they lost and how stress they earned.
8. **Comment your code!** (10 points)
9. **Proper formatting:** (5 points)
Labels are never indented. Instructions are always indented the same number of spaces. Add blank lines for readability.

Extra Credit

1. Color – 5 points

Make use of color to enhance your game. The color must be meaningful – don't just set the color at the beginning of the program.

2. ASCII Art – 5 points each for a max of 10

Use ASCII-art to make your program exciting. The ASCII-art must be meaningful and not something overly simple like:

```
=====:)
```

It's a happy worm!

You can use multiple `.ascii` directives under the same label. Only place the `\0` at the end of the final one. You will also have to put a backslash `\` before any other backslashes or double-quotes `"`. For example, the following creates a square.

Square:

```
.ascii "****\n"
.ascii "*  *\n"
.ascii "****\n\0"
```

3. Bound the percentages – 5 points

The most basic version of the program allows the student to have knowledge, stress, and endurance outside the "normal" range of 0% to 100%. For this extra credit, make sure these values stay in the normal range.

4. Zero Endurance penalizes and doesn't end the game – 5 points

When the student's endurance reaches 0 (or maybe less), the game automatically ends. Instead, make them go to the hospital wing (or some other penalty) where they lose a random (but considerable) number of days. Their endurance should also be reset to 100%.

5. At least 2 more decisions – 5 points

There is more to do than just studying, resting, and playing quidditch. Give the program more decisions that can help or hinder the game.

6. Another resource – 5 points

Right now, there are only three resources. What else can be considered? Food? House points? Happiness? Money? Love?

7. Random events – 5 each for a max of 15.

What other types of events, good and bad, can occur. The more you add, the more you can capture the feeling of being a student at Hogwarts!

Due Date

The assignment is due at the **end of Dead Week**. I strongly suggest that you get to work on this assignment as early as possible. If you did well on your labs, it shouldn't take more than a few hours.

Submitting Your Project

To submit your lab, you must 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 (do not send the a.out or the object file). Send the .asm file to:

```
dcook@csus.edu
```

UNIX Commands

Editing

Action	Command	Notes
Edit File	<code>nano filename</code>	"Nano" is an easy-to-use text editor.
E-Mail	<code>alpine</code>	"Alpine" is text-based e-mail application. You will e-mail your assignments it.
Assemble File	<code>as -o object source</code>	Don't mix up the <i>objectfile</i> and <i>asmfile</i> fields. It will destroy your program!
Link File	<code>ld -o exe object(s)</code>	Link and create an executable file from one (or more) object files

Folder Navigation

Action	Command	Description
Change current folder	<code>cd foldername</code>	"Changes Directory"
Go to parent folder	<code>cd ..</code>	Think of it as the "back button".
Show current folder	<code>pwd</code>	Gives the current a file path
List files	<code>ls</code>	Lists the files in current directory.

File Organization

Action	Command	Description
Create folder	<code>mkdir foldername</code>	Folders are called directories in UNIX.
Copy file	<code>cp oldfile newfile</code>	Make a copy of an existing file
Move file	<code>mv filename foldername</code>	Moves a file to a destination folder
Rename file	<code>mv oldname newname</code>	Note: same command as "move".
Delete file	<code>rm filename</code>	Remove (delete) a file. There is no undo.