**Capstone Projects**

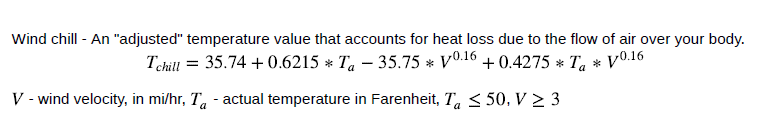
**Simon Says**

"Simon Says" is a memory game where "Simon" outputs a sequence of characters (R, G, B or Y) and the user must repeat the sequence.

1. When the user starts the program, Simon will give the character. The User will repeat the character.
2. Simon will duplicate the first character and add another. The User will repeat the two characters in the same order as Simon.
3. Simon will repeat the first two characters and add another. The User will repeat the three given characters
4. The program will continue playing as long as the User can
5. After the 5th, 9th and 13th sequence, Simon will play a sound of congratulations.
6. If you fail to repeat the sequence exactly, Simon will make a sound and exit you from the game.

**More on Temperature Conversion**

In class you took user input to convert Fahrenheit to Celsius in a function. You learned to read from an input file for a range of temperatures at Aberdeen Proving Ground and converted them to Celsius.

* Write another function to convert Celsius to Fahrenheit, read from an input file you created with temperatures for a year in Celsius from a city of your choice. Covert these into Fahrenheit and plot these by month.
* Ask the user to input which conversion they want to use and the name of the input file. Make sure you do error checking.
* Write a function that computes the wind chill, given a temperature and wind velocity and test your function on some values.
  + ****The greatest range of wind speed for effect is from 5 to 50 MPH. Take these ranges into consideration for your calculations
* Plot the Windchill. Y-axis is Wind Speed. X-axis is Temperature

**Password Strength**

Write a program that takes in a string that should represent a password and determines if the password is strong enough. A strong password is characterized by:

* Containing at least 12 characters, but no more than 16
* Containing at least one digit
* Containing at least one upper case and one lower case
* Containing at least one special character !, @, #
* If the password does not meet these requirements, tell the user and ask for another password
* If the password is acceptable, ask the user to input the password again. Compare the two, if the password does not match, ask for input again. If the two matches print “password accepted” and exit.
* Write your code as functions

**Vacation Planning**

Write a program that calculates the cost of your vacation. Query the user for the following input:

* For the dates of your vacation, the city you are going and spending money.
* Create a formatted file that gives the cost of plane fare for at least 10 cities
* Create a formatted file that gives the cost of rental car for the same 10 cities.
* Create a formatted file that gives the cost of hotel for the same 10 cities.
* Note: you can create one file that has the cost of plane fare, rental car cost and hotel costs. Think about how you are going to store the data.
* Use functions to calculate cost of plane fare, hotel and rental car.
* The program will print the total cost of your vacation that includes cost for hotel, rental car, plane flight and spending money.

**Tip & Tax Calculator**

Write a program to function as a tip and tax calculator. Ask the user for the state they are in, the restaurant they are dining at, the cost of the bill, and whether the service was “poor,” “okay,” or “great” (each which corresponds to a certain tip percentage). Calculate the correct tip, tax, and overall payment and write it to a text file named *receipt.txt* that mimics the format of a regular receipt (with information such as date, name of restaurant, cost, tip amount, tax amount, total cost, and a signature line).

Note: Be sure to also create a text file that lists states and their tax amounts. Read from that file to create a list or dictionary that you can access to correctly calculate the tax once the user indicates the state they are in and the cost of the bill.

**Monoalphabetic Cipher Decryption**

Write a program to decrypt a monoalphabetic cipher by matching the character frequencies in the ciphertext to the character frequencies in normal English text. Use the character frequencies in english.txt as a reference.

Be sure to save the key in a file so that you can modify it to get the exact key and correctly decrypt the ciphertext.

**Text-Based Video Game**

Create a turn-based text-based video game that puts you up in a fight against an enemy. To start this game, you are required to input your name. Once you input your name, an enemy will attack you and your health will go down by a small random amount. Next, you will have the choice to **attack** or **heal**. The results from attacking will be determined by a random number between 1 and 6 (the equivalent of rolling a die). Scenarios for each side of the die will be determined by you (e.g., if you roll a 1, something happens and you inflict some damage between two numbers).

After you attack, the enemy will have a chance to hit you and may land or miss their attack. Whether they hit you or miss you will be determined randomly. After the enemy attacks, the turn is over and you will be prompted once again to attack or heal.

The result from healing will be determined by a random number between 1 and 10. The enemy will not be able to heal, as doing so may result in a game that takes a while. At the end of each turn, you should display your current health and the enemy’s current health.

The game ends once your health or the enemy’s health either reaches 0 or drops below 0. In addition to the end of the game, you are required to display the remaining health of the victor and produce a .txt file that contains the following information:

Average damage dealt this run:

Maximum damage dealt this run:

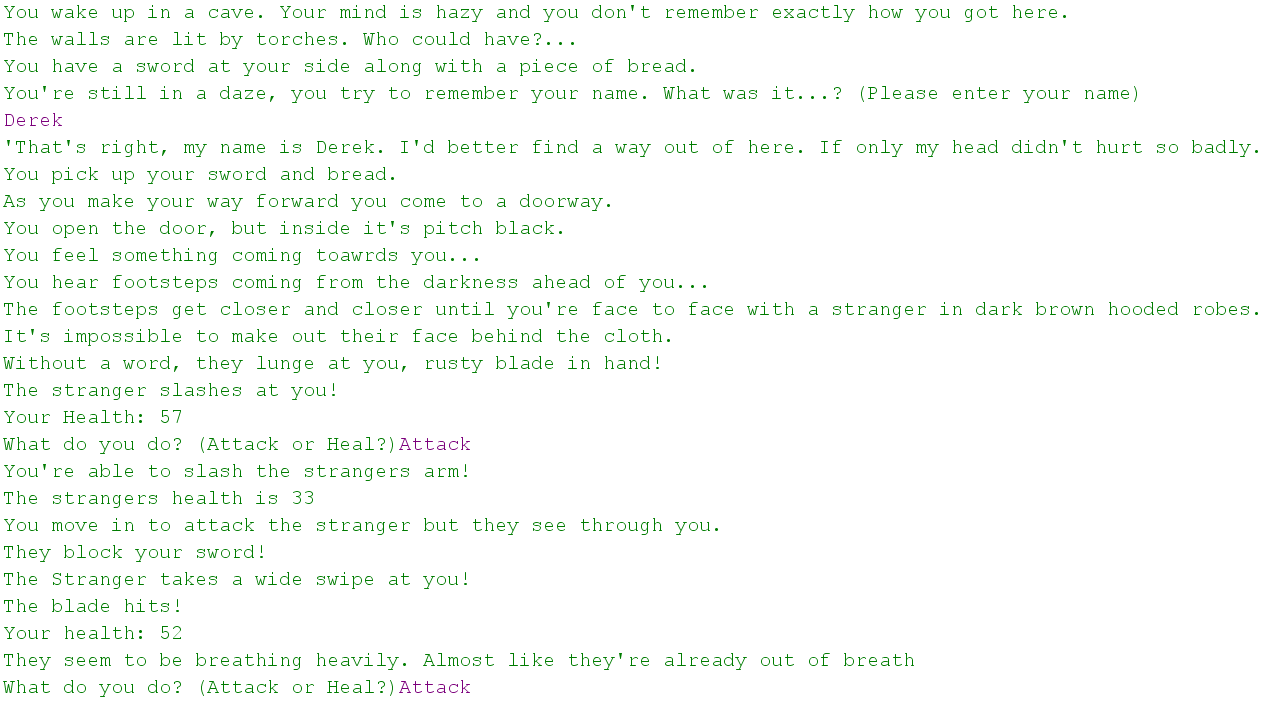
Minimum damage dealt this run:

Average damage received this run:

Maximum damage received this run:

Minimum damage received this run:

Number of turns taken this run:

Please keep in mind that in addition to writing the code to create the game, you will also have to make your own story and scenarios. Below is an example of what your program may include (the text in purple is what the user inputs):

**Generating a Histogram**

Create a histogram-generating tool which reads a file (“midterm.txt”) containing students’ midterm scores associated with their ID numbers and names. The data are to be saved in a list called “names” and a list called “scores.” The file “midterm.txt” includes the following:

1 Adam 85

2 Benjamin 90

3 Brian 91

4 Casey 98

…

The program should find the total sum, average, maximum, minimum scores and have the records displayed with a character-based histogram. The histogram may be classified into score ranges divided into 6 levels from the minimum to maximum score. The program output will be:

71 ~ 75 : / (1)

76 ~ 80 : //// (4)

81 ~ 85 : ///// (5)

86 ~ 90 : ///// (5)

91 ~ 95 : // (2)

96 ~ 100 : /// (3)

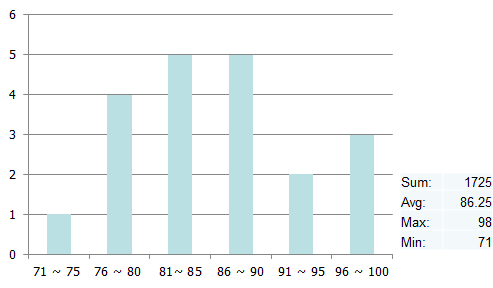
Sum : 1725

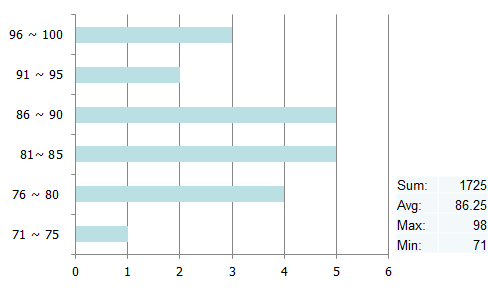
Avg : 86.25

Max : 98

Min : 71

After generating a character-based histogram with the total sum, average, maximum, and minimum scores, create a graphical histogram using matplotlib with the same information. Be sure to include a title and labels for the axes. Also, those who have above a 90 should have a green bar while those who scored below have a default-colored bar





**Projectile Programming Project**

**Horizontal Projectile**: Simpler project

1. Equations:

x = (vx)(t)

y = ½(a)(t)2

1. Problems to solve depending on your programming ability and time available:
2. Problem 1

Given y and vx, Calculate x

1. Problem 2

Given y, divide the problem up into time segments, and calculate x and y to create a firing table of those values over the path of the projectile

1. Problem 3

Plot the path using matplotlib in Python

Edit the program to make it user friendly by including and checking user input prompts and possibly including a menu option to allow for different types of solutions depending on user choice.

**Angle Projectile**: More complex project

1. Equations:

vx = (v)(cosѲ)

vyi = (v)(sinѲ)

x = (vx)(t)

y = (vyi)(t) + ½ (a)(t)2

1. Problems to solve depending on your programming ability and time available:
2. Problem 1:

Given v and Ѳ solve for the projectile range and time in the air assuming y = 0.

1. Problem 2:

Given v, Ѳ and y solve for the projectile horizontal distance and time in the air to that altitude (y). Check for all possible results based on the givens.

1. Problem 3:

Given the x and y position of the target, determine the required v and Ѳ needed to hit the target.

1. Problem 4:

Given y, divide the problem up into time segments, and calculate x and y to create a firing table of those values over the path of the projectile

1. Problem 5:

Plot the path of a projectile using matplotlib in Python.

1. Problem 6:

Given v, determine how to aim the launcher in order to hit a moving target that your program generates. Plot the motion of both objects to see if their paths intersect.

Edit the program to make it user friendly by including and checking user input prompts and possibly including a menu option to allow for different types of solutions depending on user choice.