To: Professor Mukasa

From: Nick DePatie, GE1502SEC23

Date: February 4, 2021

Subject: Mission #2, Blackbird Reconn V2

Attachments: 2

For this mission, I was tasked with creating a program that calculated the shortest route in between 20 different coordinates through random generation and creating a dual authentication cipher system to access the program.

How the Code Works

To plan out how I wanted to organize and implement the code, I began creating a flowchart (attached below) to achieve the listed objectives in the given problem.

The first objective was creating a cipher that only the correct user, Captain Molly, would have the key to. I did this by using a preselected seed to generate a "random" number that only the user would know. I also needed to implement a dual authentication system where the program asked the user what the name of the classified mission was. If the user correctly inputs both (1 and LADYBUG, respectively), then the program would move on to the second objective.

The second objective was to process the coordinates that were in a given file and save them to arrays. I did this by using a for statement and only storing information in one row at a time. Once the row was filled (i.e., both x and y coordinates were stored), the statement would repeat for the next row. This would repeat 20 times to save all the coordinate data to an array.

The third objective was to calculate the distance of a randomly generated route. The user is asked how many routes they would like to generate, and the calculation and generation will happen however many times the user inputs. The route is randomly generated through a shuffle command that shuffled an array of number 0-19. The distance between each location is then calculated using the distance formula derived from the Pythagorean Theorem using the randomly generated order using a for statement that repeats until all locations have been used. These distances are all summed up and compared against the shortest distance generated so far. If the new total is less than the shortest distance, then that total becomes the new shortest distance. Once the calculations have been repeated the desired number of times, the shortest distance is outputted to the user.

The fourth objective was to output the order of locations and the shortest distance to a new file. I did this by saving the order that was used to generate the shortest distance every time a new shortest distance was calculated. I then formatted a table in a new file using input output manipulation code and inputted the data using a repeating for statement to input all the values in a certain row, then, once the row has finished printing, the next row would start printing.

Testing to Make sure the Code Works as Intended

While coding, I would occasionally run into some hiccups that would prevent me from continuing. It would be that I forgot to define a variable globally instead of locally, or that I messed up the syntax a bit. That would typically be easy to fix, as the code would not run when those mistakes were present.

However, I ran into one problem that I was having trouble solving. I was experiencing a stack overflow error within the code that stopped my code from running, even though it was syntactically correct. I worked out the problem by placing "cout" statements occasionally in the code to debug and find the root of the problem. Eventually I found out that it was in the calculation portion of my code and, after testing almost every line within my function, I figured out it was that I put a "b" variable instead of an "a" variable because I mixed up my for statements.

I also originally added a cout statement within the if statement testing to see if the calculated distance was less than the shortest route so far to see when the program would get a new short route. When I finalized the code, I removed this, since the original problem asked for the distance to not be repeated over and over in the console.

Plot and Prediction

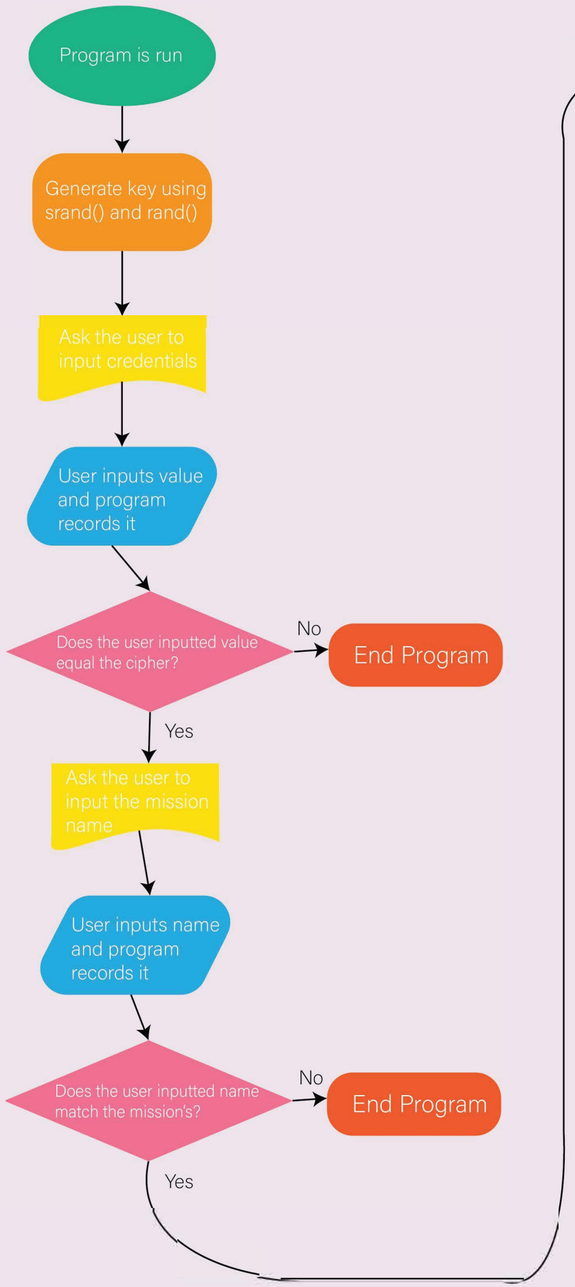
I was also tasked with creating a scatterplot and line of best fit for the data that was outputted from my program. I plotted the length of the shortest route found vs the number of routes the user asked to calculate. I inputted this data into excel (attached below) and found a line of best fit when I took the log of the number of times the route was generated. The line of best fit can be represented by the equation:

$$y = -21.89\ln(x) + 908.9$$

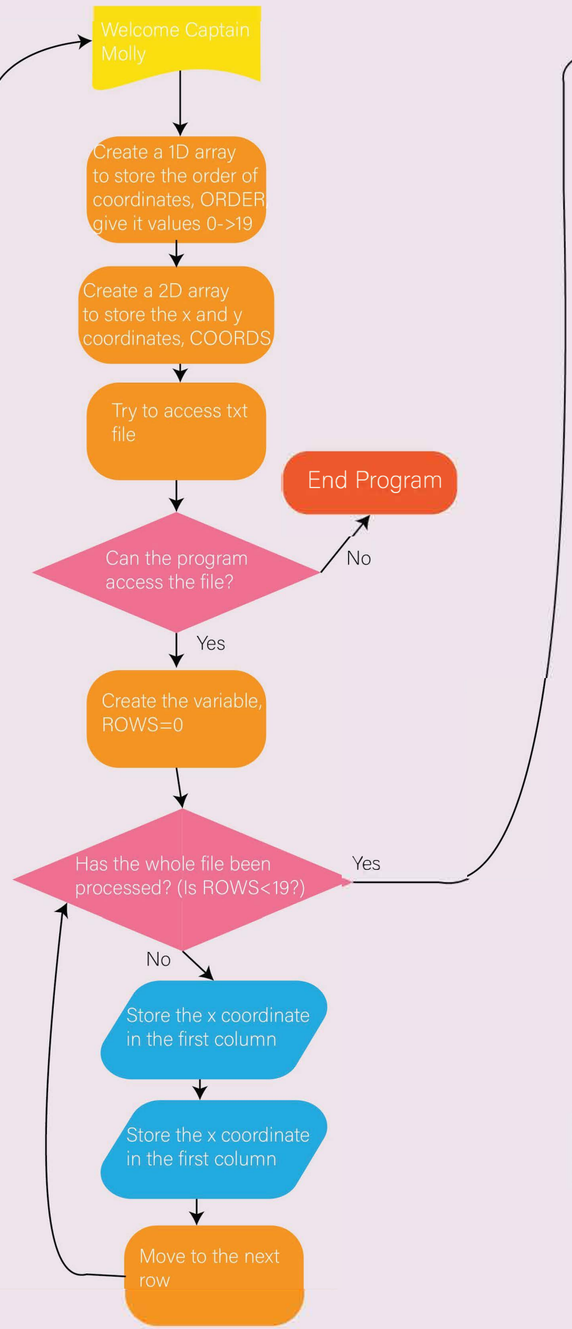
(With y being the shortest route length and x being the number of generations)

Therefore, if I were to find the approximate length of a route generated 1,000,000,000 times, it would be **455.2677 miles long.** This number makes some sense looking at the general trend of the graph, but, after how many iterations of the data I have seen, it seems unlikely the total route length would reach less than around 490 or 480 miles long.

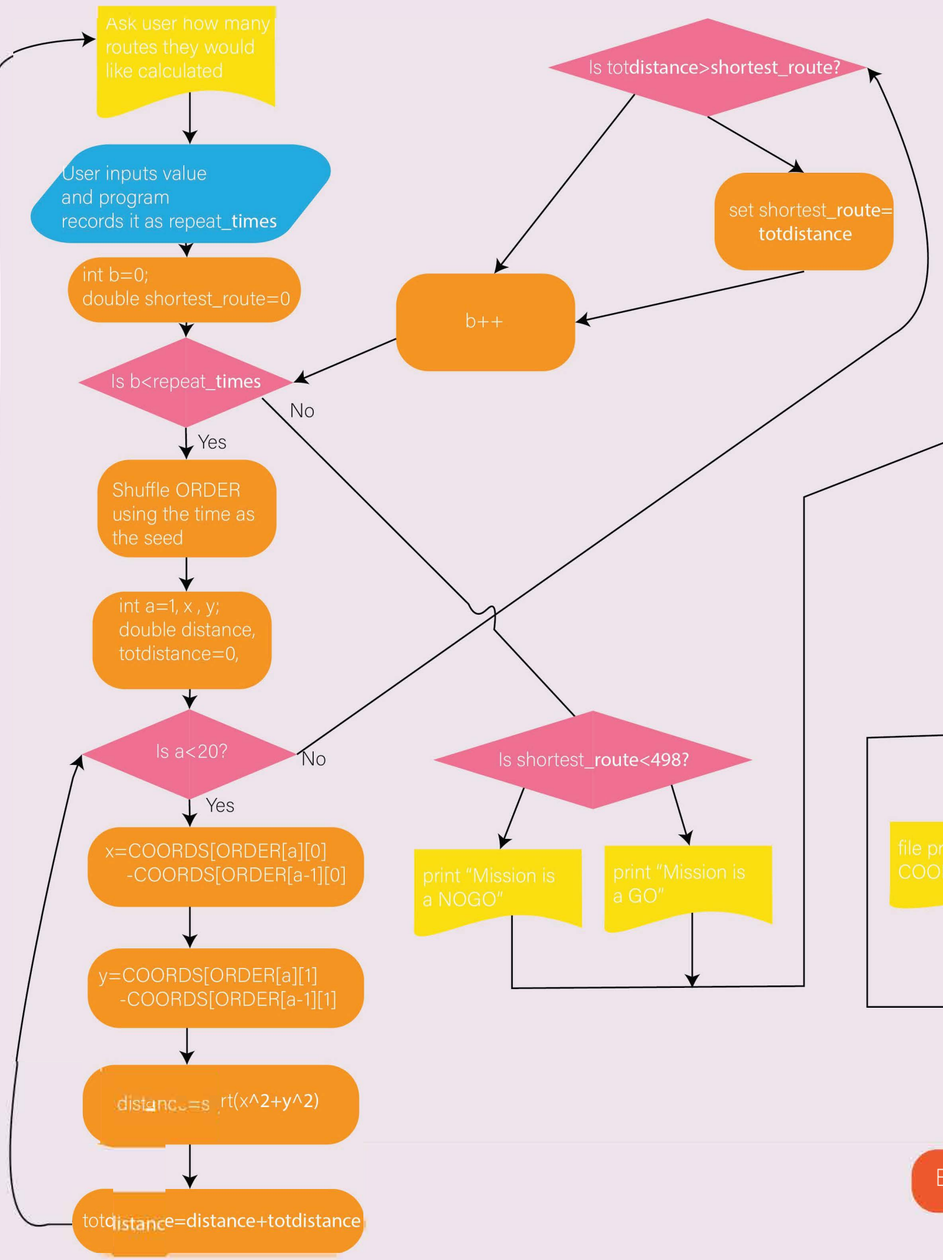
Cipher



File Processing

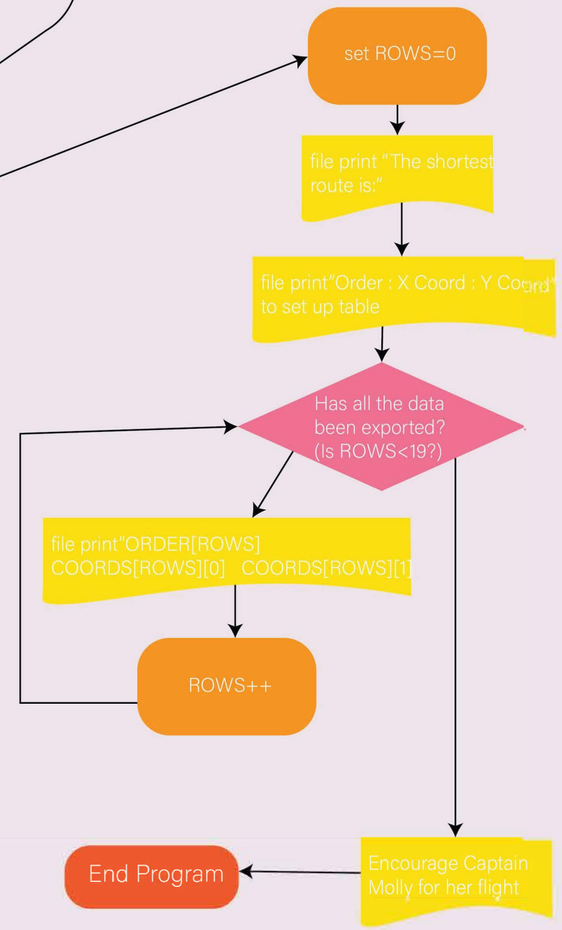


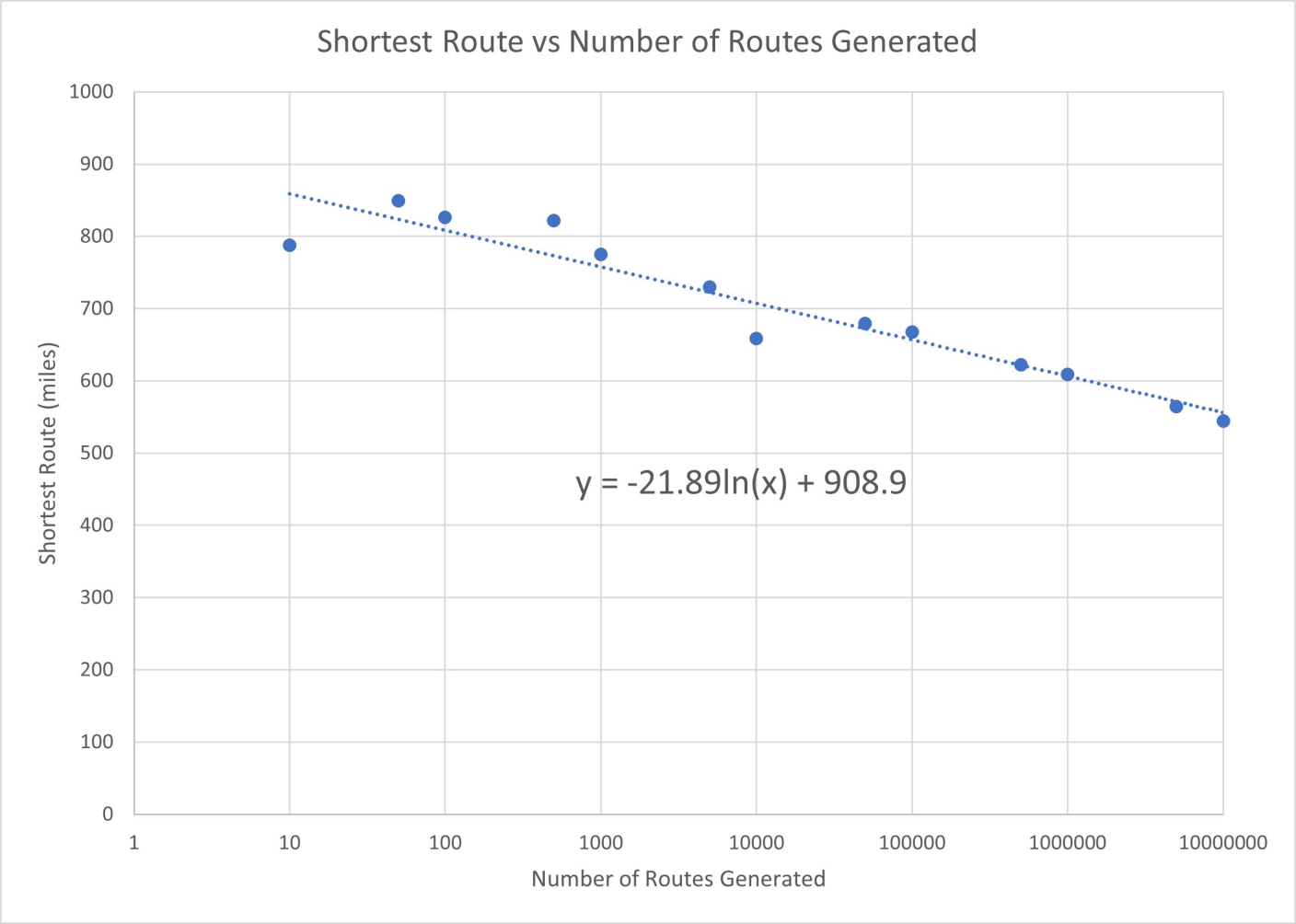
Route Calculation



Route Analysis

Exporting Data




Shortest Route vs Number of Routes Generated
Shortest Route (miles)
1000
900
800
700
600
500
400
300
200
100
0
y = -21.89ln(x) + 908.9
1
10
100
1000
10000
100000
1000000
10000000
Number of Routes Generated