**Project 1**

Jeremy Boyd

[Computer Science and Engineering]

Speed School of Engineering

University of Louisville, USA

[jrboyd03@louisville.edu](mailto:jrboyd03@louisville.edu)

1. **Introduction** (What did you do in this project and why?)

For this project I was responsible for solving the Travelling Salesperson Problem which is a problem programmers have been working on since forever. It involves a salesman being given a list of cities to travel to and their coordinates the problem is to find the most efficient route.

The approach I took to solve this problem was the brute force approach by generating all the permutations for possible routes to travel through each city with no backtracking. And then calculate the total distance for each route and figure out which route would be the shortest and most efficient. This approach solves the problem because the program I created will literally generate every possible route through all the cities. And calculate the total distance for every single route so it’s full proof in terms of being able to find the shortest route although it’s not full proof for larger data sets as it’s just not very efficient. And with larger data sets will probably cause your computer to crash.

1. **Approach** (Describe algorithm you are using for this project)

The programming language I used was Python for this project. I attempted many different approaches when trying to come up with a solution for this project. But to start I imported the math, itertools, pandas, and numpy libraries. Then I created a function called ‘distance’ to calculate the distance between two points using their x and y coordinates. The next thing I did was use the ‘pandas.read\_table’ function to read in the data from each tsp file and store it in a table. After this I generated a list that held all the x and y coordinates stored in the table that was read in and I’ll explain its purpose later on in the paper.

After that I used a function called ‘itertools.permutations’ which allowed me to easily generate all the permutations for the number of cities and store them in a list called ‘allPaths’. Using a range that went from 0 to a count variable I created that held the number of cities from the tsp. Next, I created a list called ‘distances’ that would recursively go through the list of coordinates I mentioned earlier and find the distance between each set of points. Then in conjunction with an asarray that I created using the numpy library. All the values that are stored in the ‘distances’ list are imported into the array along with their indexes so that the asarray functions as a sort of look up table.

Next I created a set of nested for loops where the outer for loop will cycle through each permutation in the ‘allPaths’ list and the inner for loop will go through the numbers in each individual permutation. By looking up the distance between each edge in the permutation in the asarray and adding the returned results in a variable. Once the inner for loop has finished a cycle it exits and the value in the variable is stored in a list called ‘distanceTraveled’ then the outer loop restarts and sets that variable back to 0. And this continues until the loop has gone through all the permutations.

Lastly, I created a variable that returned the smallest value in the list and another variable that uses ‘allPaths’ and the corresponding index in ‘distanceTraveled’ that matches the smallest value in order to find the route that goes with the shortest distance. And printed those both on the screen.

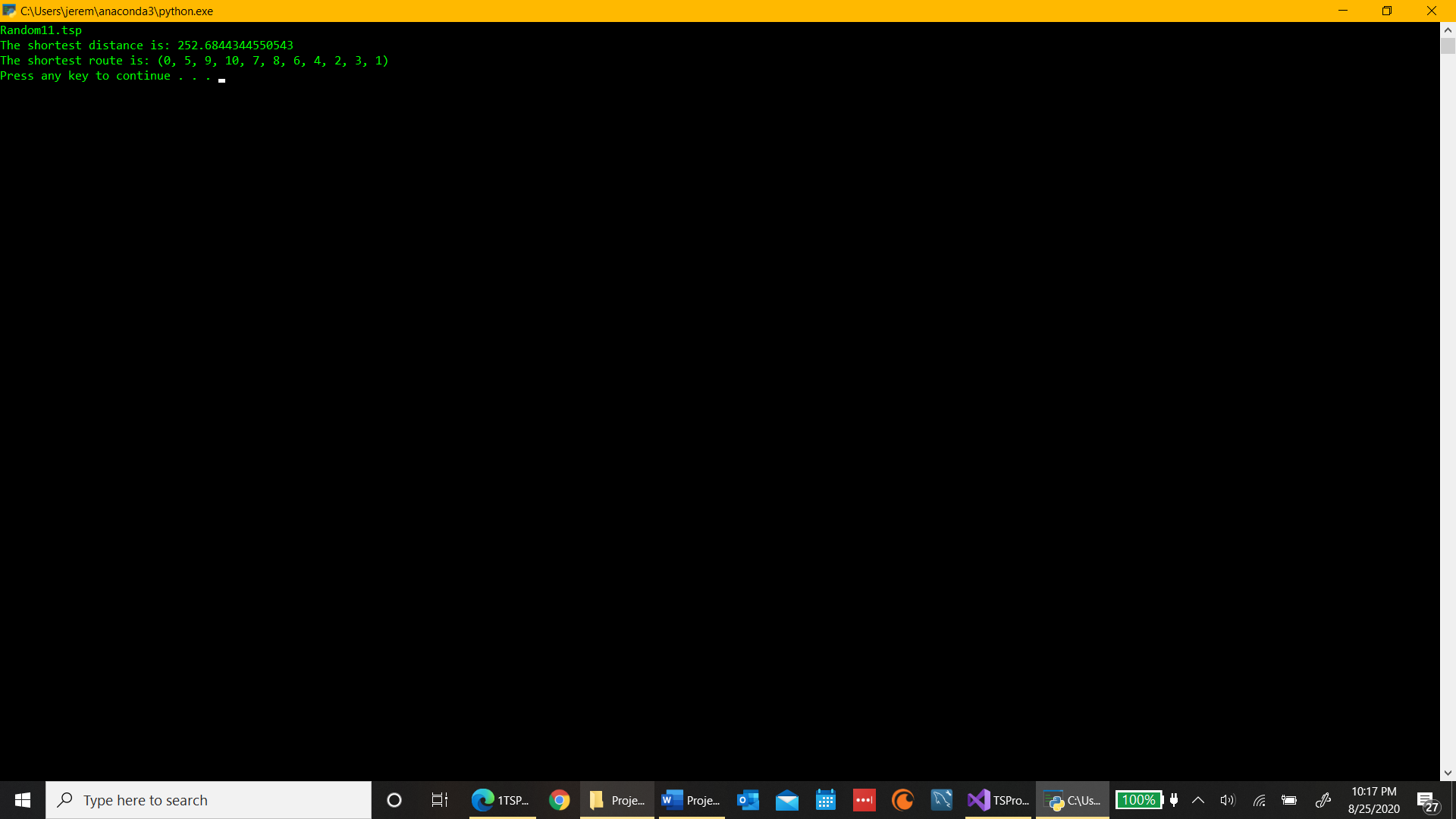
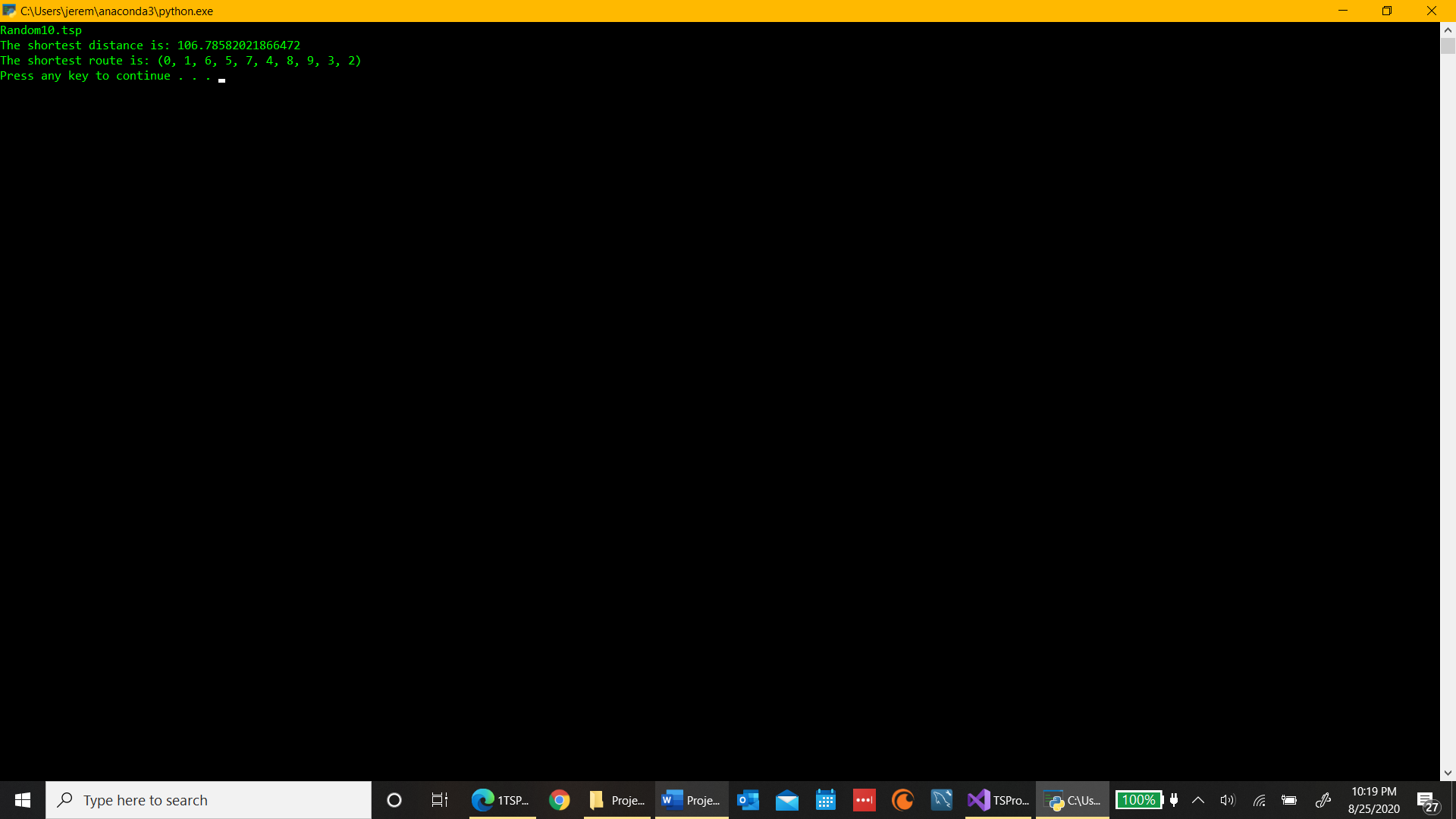
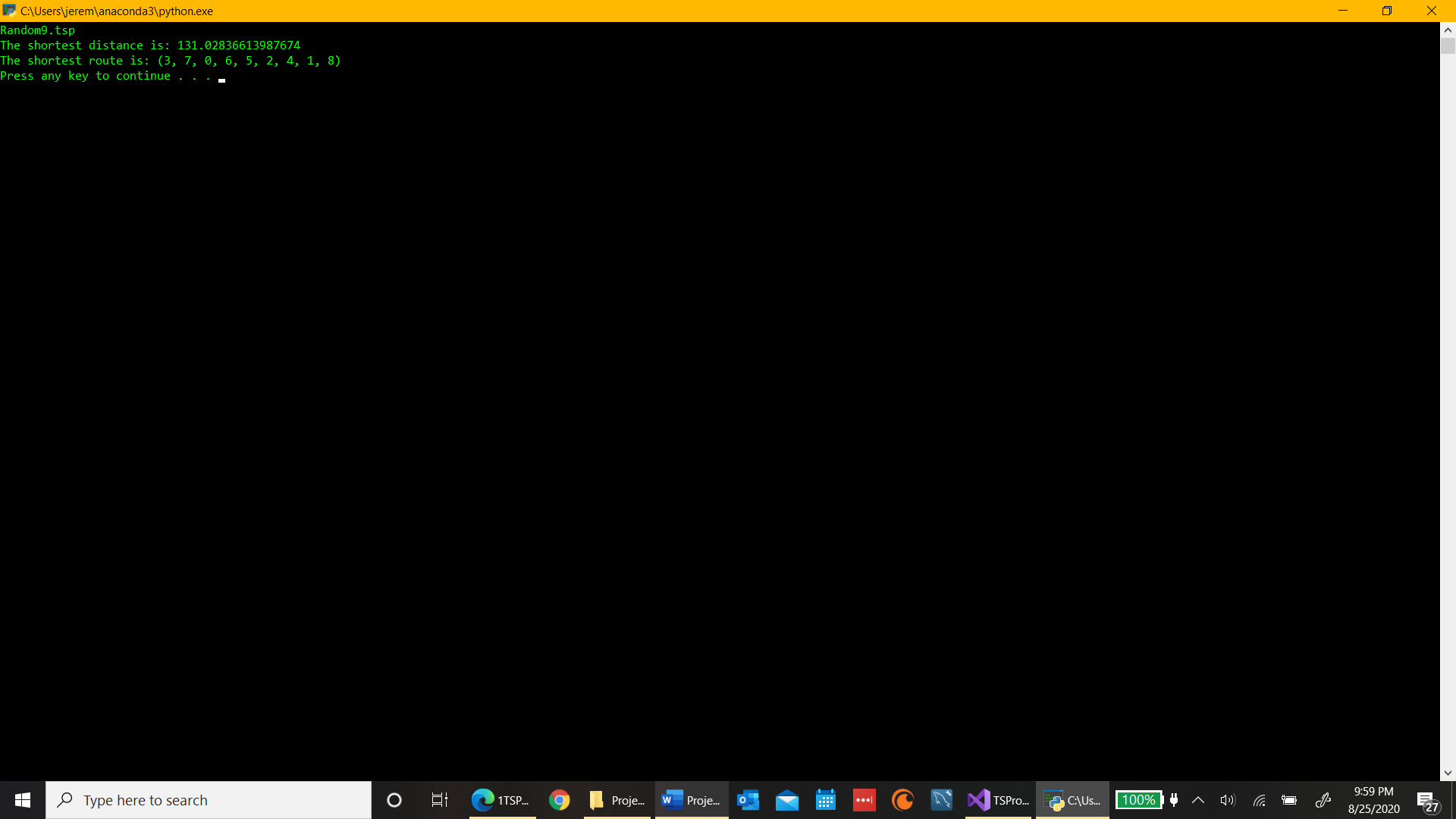
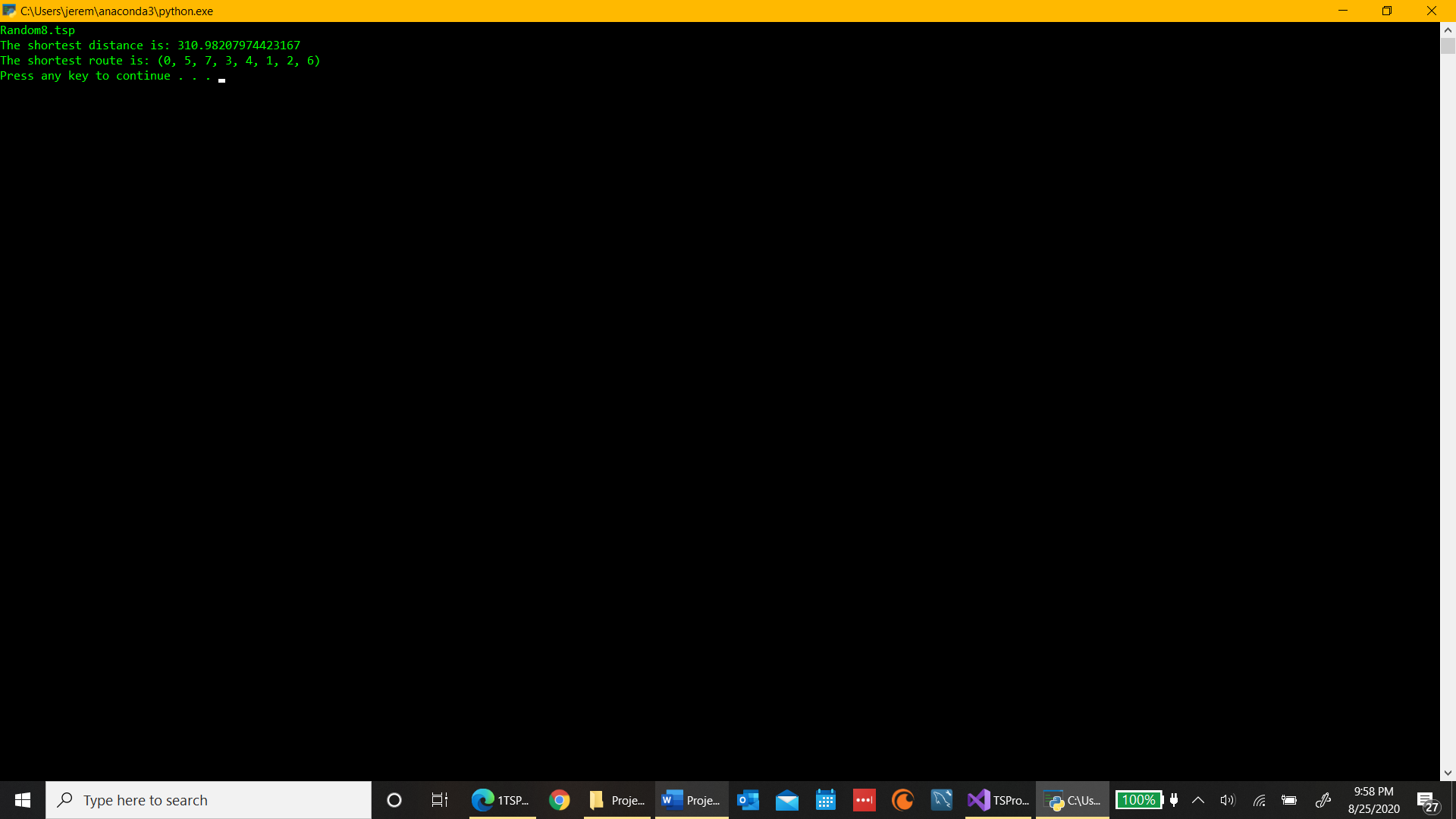
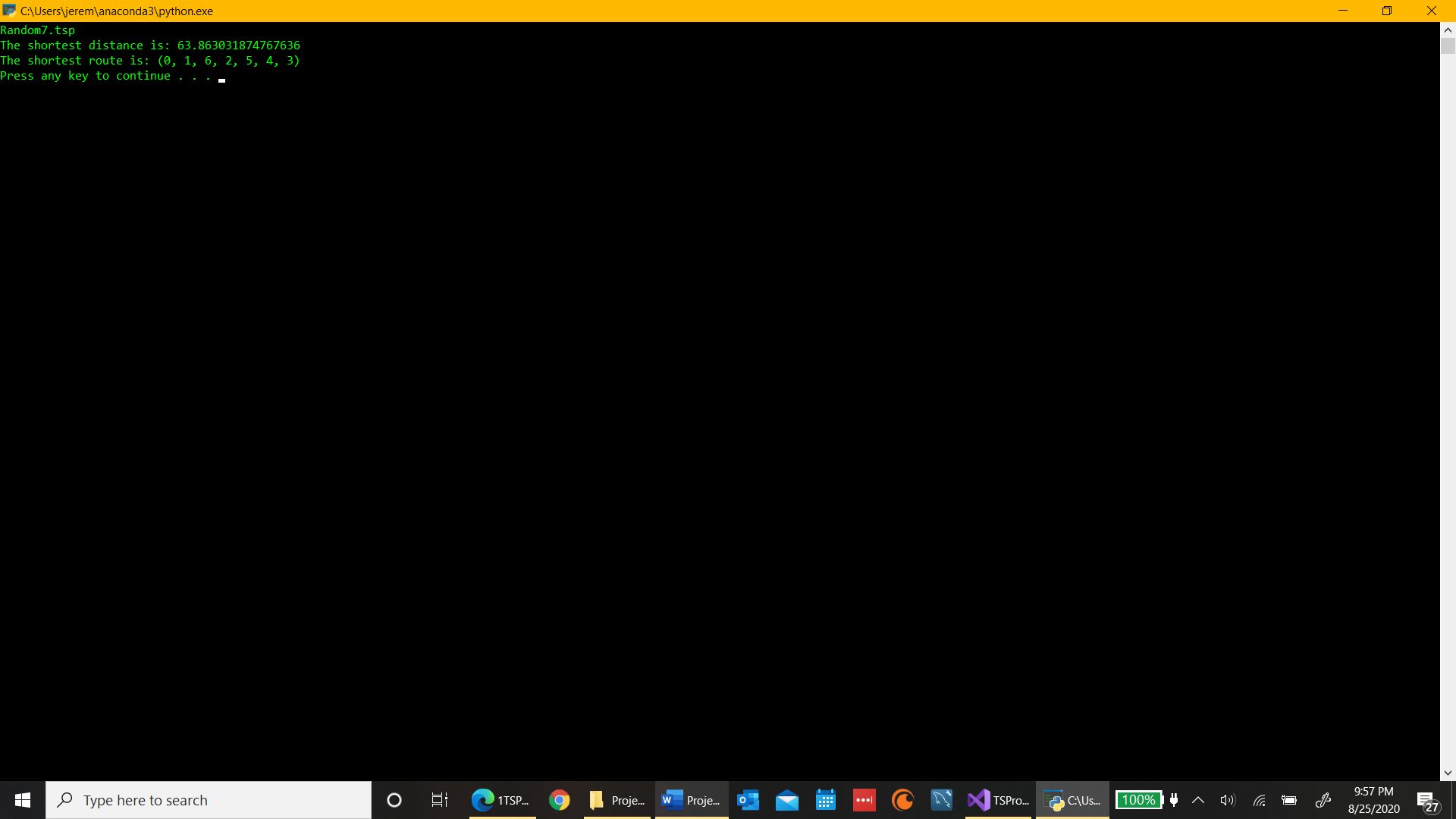
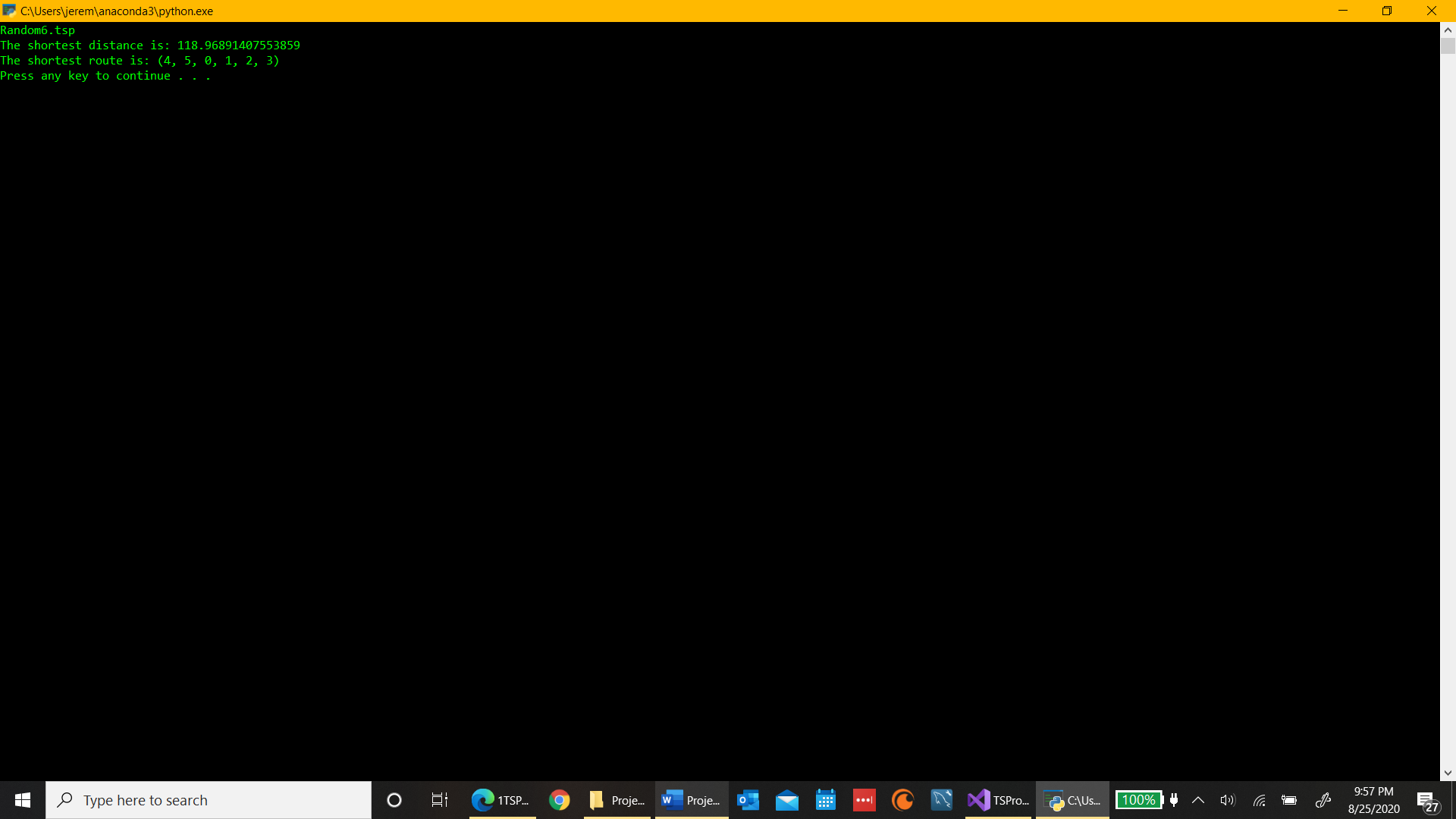
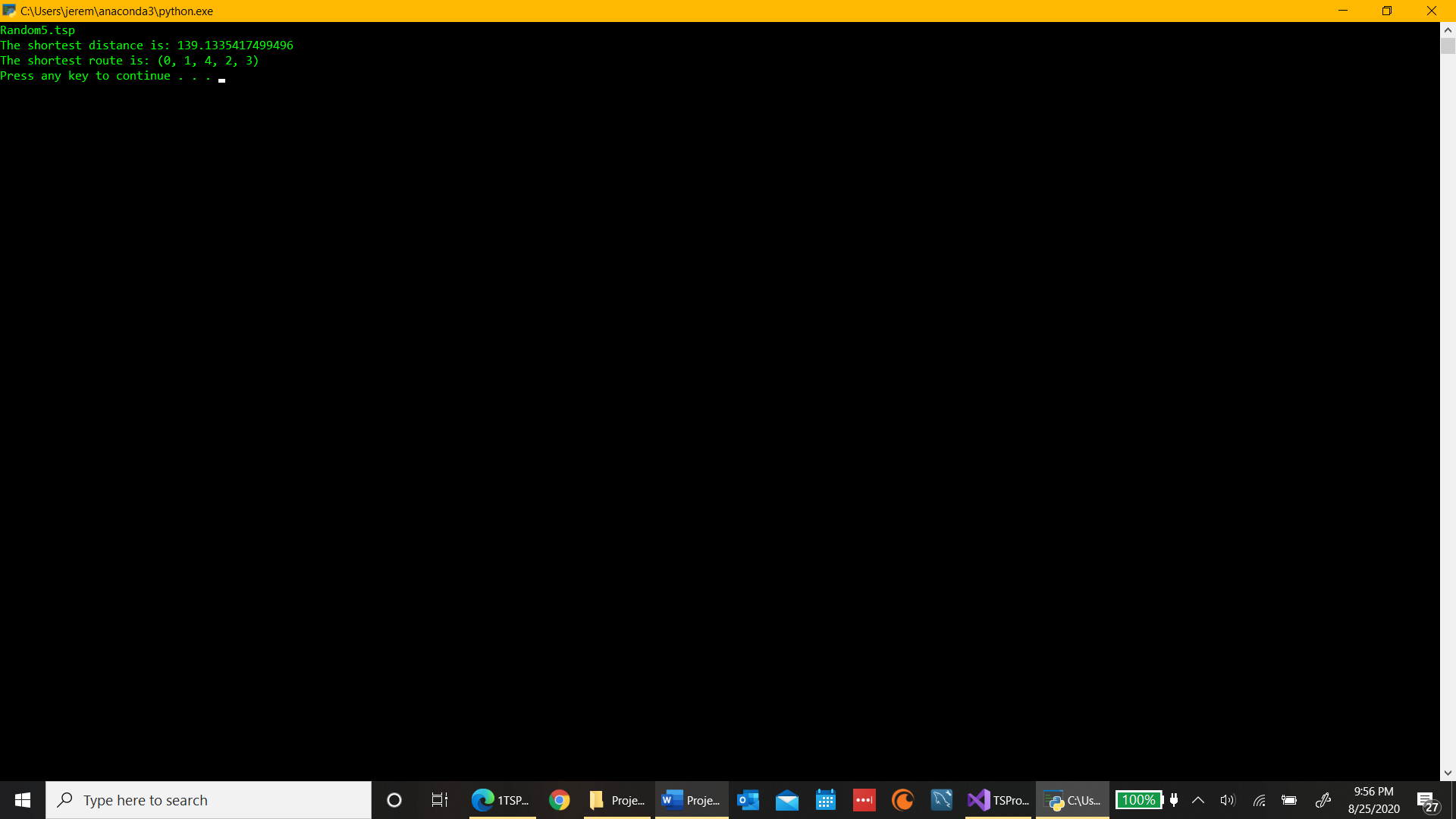
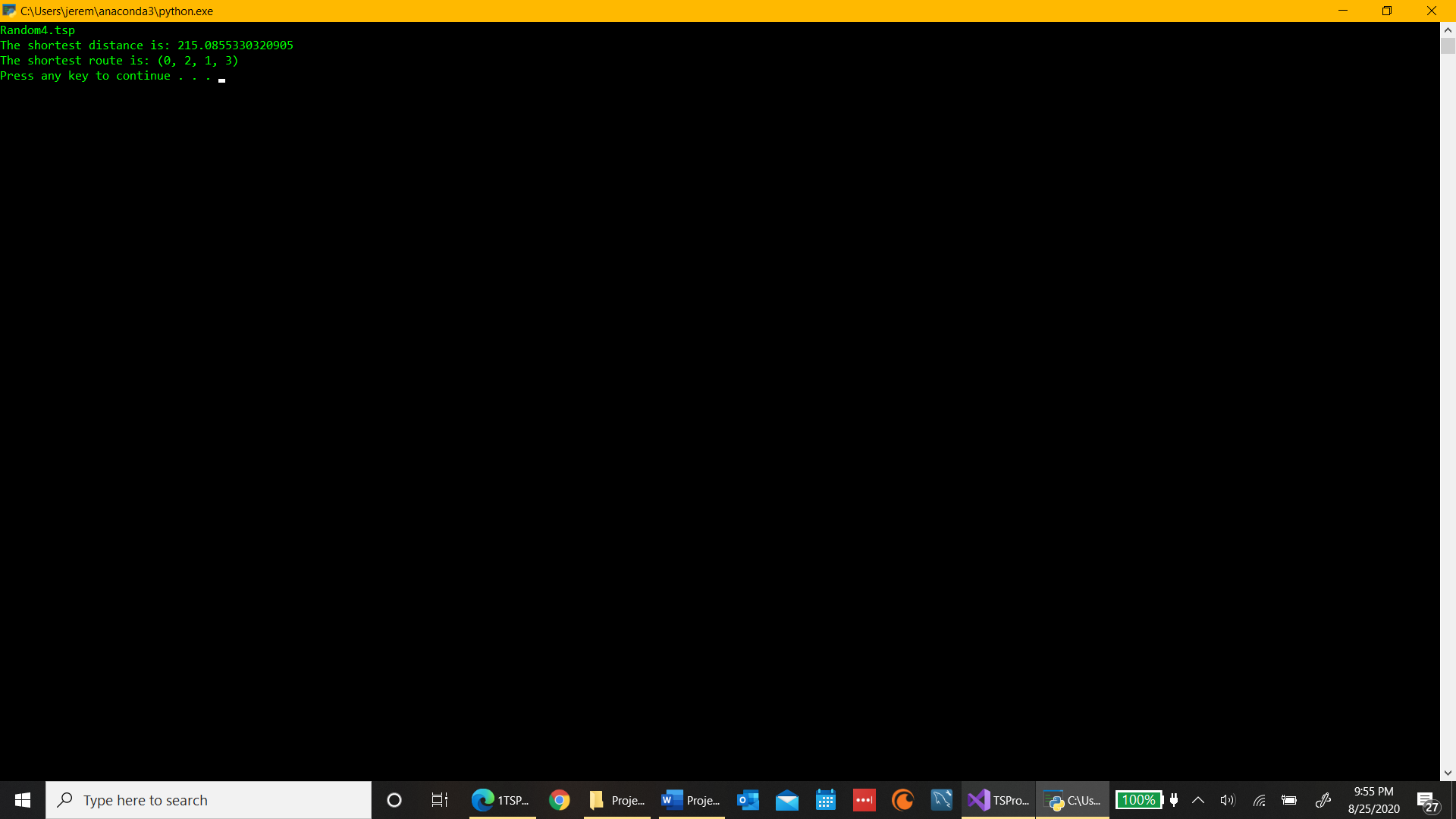
1. **Results** (How well did the algorithm perform?)

The algorithm performed well for 1-9 cities those all took less than 10 seconds. Once I got to 10 the performance started to dip off as it took me a little over a minute with 10 cities. And then for 11 cities it took 13 minutes and for 12 cities I couldn’t even get a result cause the program would crash.

* 1. **Data** (Describe the data you used.)

I just used the data in the multiple tsp files we were given, which included a number as a city and its respective x and y coordinates .

* 1. **Results** (Numerical results and any figures or tables.)



1. **Discussion** (Talk about the results you got and answer any specific questions mentioned in the assignment.)

I was able to get the shortest distance along with its corresponding route for all the given tsp files and any other random numbers I put in. And for small data sets the algorithm worked well. But once I got to 11 cities it took a long time and at 12 the program crashed.

1. **References** (If you used any sources in addition to lectures please include them here.)

*Traveling Salesman Problem*, www.math.uwaterloo.ca/tsp/.