# Assignment 1

## Instruction

There is 1 question in this assessment, and it will be evaluated according to the **correctness, performance,** and **documentation.** You should describe your algorithm design as detail as possible in your documentation. You may want to include UML diagrams such as sequence diagram to describe your algorithm.

Please submit before **Monday of Week 7**. Late submission and plagiarism will not be tolerated.

Q1. Given a set of locations, find the **minimum distance** of tour that visits each **red** point exactly once using brute-force method with **the constraints that before travelling to each of the red point, a green point must be visited first. The total distance, including the green point must be the shortest. If there are more green points than the red points, ignore them.** Write a **sequential** program to solve the given problem. Correctness is the priority and speed are the secondary objective; no performance marks will be given if the correctness of the application is not achieved.

Marking Rubrics:

| Criteria | Descriptions | Marks Distribution |
|---|---|---|
| **Correctness** | Did the program interoperate properly with the given program? | **15%** |
| | Did the program produce correct output? (visit all red points) | **20%** |
| | Is the program robust? (No error after multiple consecutive execution with different data set) | **10%** |
| | | |
| **Performance** | Did the program perform its task within the required timeframe? (Minimum requirement, to be decided. Depending on available resources) | **10%** |
| | How fast the program produces all the outputs? (Compare with peers) | **20%** |
| | | |
| **Documentation** | Did the report explain the work clearly? | **5%** |
| | Did the report consist of model that describe the program clearly? | **5%** |
| | | |

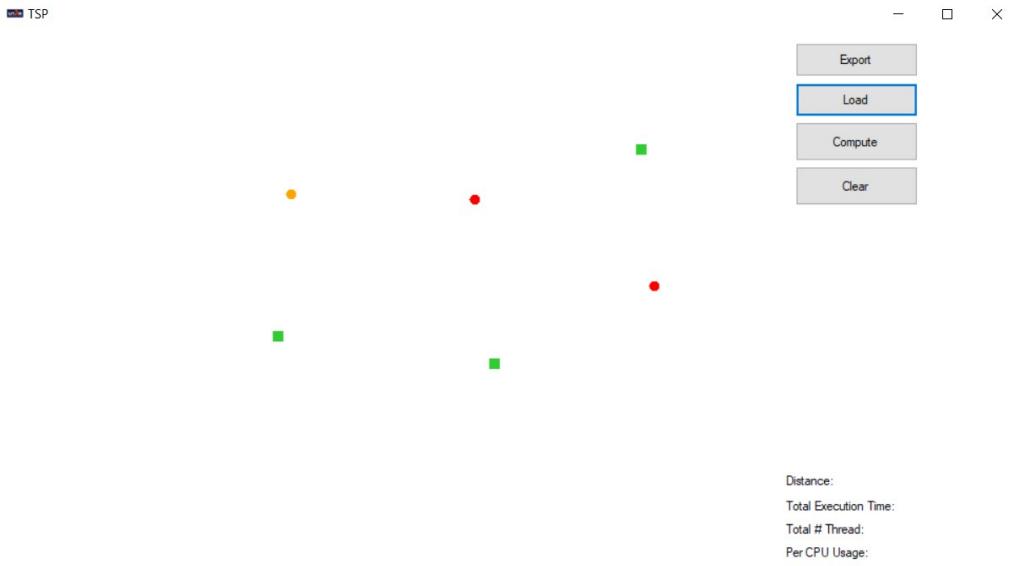
| | | |
|---|---|---|
| **Code** | Programming Clarity (comments) | **5%** |
| | Programming Structure (arrangement and functions) | **5%** |
| | Programming Style | **5%** |
| | | |
| **Total** | | **100** |

* Each element of non-compliance will be penalized with respect to its severity.

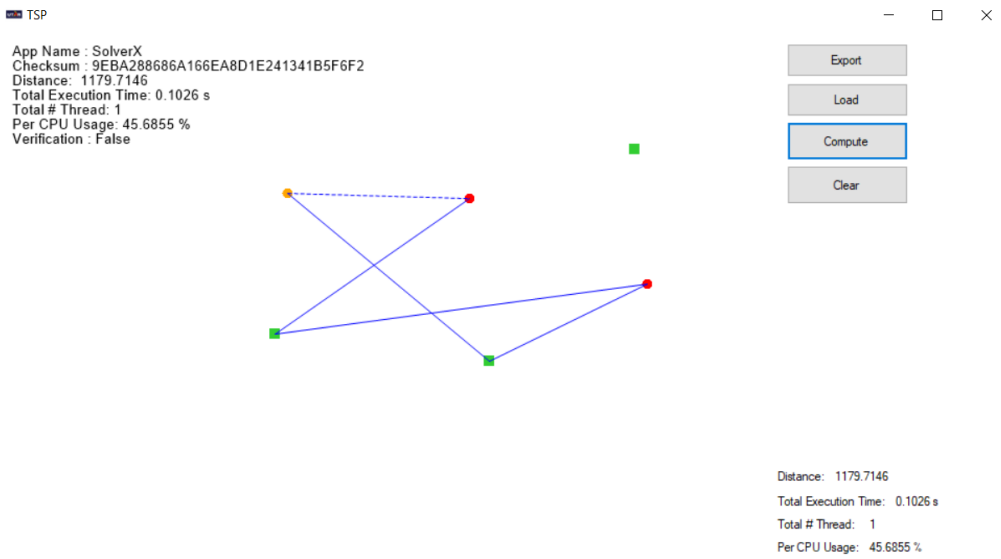
## Instructions:

1. Download and unzip the TSP program from WBLE.
2. Run the program by executing the TSP.exe. The following is the screen shot of the TSP application.

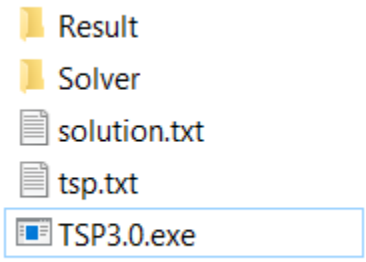

3. Test the program installation by clicking on "**Load**". You should see something similar to the figure below.




4. Click “Compute” to test the application and wait the application to complete. You should see the following screen shot. Please note that the default solver will not produce correct answer.

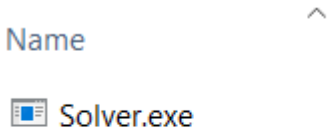


5. Open the TSP folder in your computer. You should see the following files and folder.



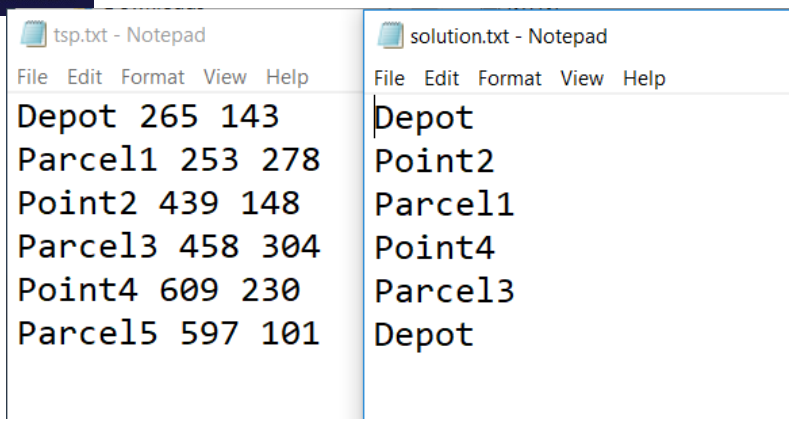

Note:
i) TSP.exe is the application to test your solution
ii) Do not delete the files and folders.

6. Open the TSP\Solvers folder in your computer. You should see the following files. You will see the following executable.




Note:
i) The Solver.exe is a dummy program. It does not solve the problem correctly.
ii) Your mission is to write your own Solver.exe. Please read the following steps.

7. The **tsp.txt** and **solution.txt** can be open using MS notepad. The tsp.txt stores the information of the points and solution.txt is the solution (generated by your program). In the **tsp.txt** file, the first line is the number of points (aka. cities). The subsequent lines are the available points (cities) where the first column is the x coordinate and follows by its y coordinate. The distance between two points is the distance needed to travel between the points.




8. The solution.txt is generated by the **Solver.exe** (see no 6.).

9. You must write a C/C++ program that reads tsp.map and compute the minimum distance to complete the tour of all the points/cities. Your program must generate the solution.txt for the TSP program to verify and display the solution.

10. Compile your solution and replace the **Solver.exe** in the Solvers folder and click compute.

11. IMPORTANT! Your program does not need to compute the time taken and MUST NOT have system ("PAUSE") or any code that require user intervention.

12. You can generate your own problem by clicking on the TSP form. First click is the starting and ending point (Depot), and subsequent right clicks are the cities. Please note that left click will generate a green point (aka. parcels).

13. Please notify me immediately if you encounter any bugs/flaws with the TSP program. No marks will be awarded for hacking/exploiting loopholes of the TSP program.