The Travelling Salesman Problem (TSP) is a Central Problem in Combinatorial Optimization which can be solved using various constructive and improvement heuristics.

In this assignment, you are to implement The Nearest Neighbour Heuristic (NNH) and Savings Heuristic (SH) which are constructive algorithms. Also, the k-opt iterative improvement algorithm is to be implemented by you (for k = 2).

The tasks:

In all datasets, the N points represent specific locations in some city or country. There are n pairs of (x, y) co-ordinates in the data file.

Name of datasets	No of locations	Optimal tour cost
burma14	14	3323
berlin52	52	7542
st70	70	675

Task-1:
Run the greedy_simple version of the NNH and SH constructive algorithms with different locations or vertices as starting location. For example- for burma14, we may start with anyone of the 14 locations. For this task, find out which location gives the shortest cost tour. Also, report on the costs of average, best, and worst cases. It is possible to have 14, 52, and 70 cases for these three datasets. Use a restricted number of k = 5 cases and pick these cases randomly.

The greedy_simple results						
	Avg. cases		Best Case		Worst Case	
	NNH	SH	NNH	SH	NNH	SH
burma14						
berlin52						
st70						

Task-2:
After finishing the task-1, we know which is the best starting location for NNH and SH constructive algorithms. For this starting location, construct TSP tours using the greedy_randomized version of the NNH and SH constructive algorithms. In the greedy randomized version, you are to choose any of the k-best neighbors, instead of choosing the best neighbor as was done for the greedy_simple version. For reporting the result, use a restricted number of k = 5 which means keep 5 best neighbours to choose for moving. Also, run n = 10 cases and report the avg., best and worst cases. Since you are using a randomized algorithm, in each case the result would be different.

The greedy_randomized results						
	Avg. cases		Best Case		Worst Case	
	NNH	SH	NNH	SH	NNH	SH
burma14						
berlin52						
st70						

Task-3:
After finishing task-2, pick up the best 3 tours for each of NNH and SH constructive algorithms. Now run 2-opt iterative improvement algorithm for each of them and report the result for avg, best and worst cases out of these 3 cases.

You can implement for best improvement and for first improvement versions. For best improvement version, we select the best neighbor out of all neighbors for moving. On the other hand, we move to the first neighbor which shows improvement over the current solution, as soon as it is found in first_improvement version.

The 2-opt results for best improvement						
	Avg. cases		Best Case		Worst Case	
	NNH	SH	NNH	SH	NNH	SH
burma14						
berlin52						
st70						

The 2-opt results for first improvement						
	Avg. cases		Best Case		Worst Case	
	NNH	SH	NNH	SH	NNH	SH
burma14						
berlin52						
st70						