Question-4 [15 Points]:

Given a set of locations and distances between them, the goal of the Traveling Salesman Problem (TSP) is to find a shortest tour that visits each location exactly once. We would like to solve the TSP problem using a greedy hill-climbing algorithm. Each state corresponds to a permutation of all the locations (called a *tour*). The operator *neighbors(s)* generates all neighboring states of state s by swapping two locations. For example, if s = <A-B-C> is a tour, then <B-A-C>, <C-B-A> and <A-C-B> are the three neighbors generated by neighbors(s). We can set the evaluation function for a state to be the total distance of the tour where each pair wise distance is looked up from a distance matrix. Assume that ties in the evaluation function are broken randomly. (Note: We don't consider a tour as returning to the start city.)

a) [3 Points] If you have *n* locations, how many neighboring states does the *neighbors(s)* function produce?

for each state, could generate following number of neighbours.

$$(n-1) + (n-2) + (n-3) + \cdots + 2 + 1$$

$$= \frac{\left(1 + (n-1)\right) + (n-1)}{2}$$

$$= \frac{n^2 - n}{2}$$