Assignment 1

Vaibhav Agarwal(B15139)

Q1. I used Gale Shapley Algorithm for implementation of this problem. The runtime complexity of this algorithm is O(n^2). This is due to the fact that in worst case, every man will propose to every woman in the list of women. In other words, each time through the loop a man proposes to a new woman. There are only n^2 possible proposals and hence the loop terminates after n^2 iterations. Men propose to women in decreasing order of preference. Once a woman is matched, she never becomes unmatched; she only gets better preferences.

Pseudo Code :

```
function stableMatching {
    Initialize all m ∈ M and w ∈ W to free
    while ∃ free man m who still has a woman w to propose to {
       w = first woman on m’s list to whom m has not yet proposed
       if w is free
         (m, w) become engaged
       else some pair (m', w) already exists
         if w prefers m to m'
            m' becomes free
           (m, w) become engaged
         else
           (m', w) remain engaged
    }
}
(source : https://en.wikipedia.org/wiki/Stable_marriage_problem)
```

Q2.

I used DFS to detect cycle in an undirected graph in O(n+m) time. The O(m+n) complexity comes from the fact that DFS would traverse the whole graph until a cycle is found.
Using DFS traversal on the given graph , if for any visited vertex ‘v’, there is an adjacent ‘u’ such that u is already visited and u is not parent of v (that is to say there is a back edge between u and v), then there is a cycle in graph. If we don’t find such an adjacent vertex for all possible vertex ‘v’ , we say that there is no cycle. The assumption of this approach is that there are no parallel edges between any two vertices.

Q3.

I used BFS for implementation of this problem. All the buildings could be considered as nodes in the graph and the close buildings as having an edge between them. So number of jumps required to reach the destination will be the distance between source and destination nodes .The time complexity for BFS is O(m+n), where m is number of edges and n is the number of nodes. The O(m+n) complexity comes from the fact that BFS would traverse the whole graph until the destination is found. So in worst case if the destination node lies in the last layer and is the very last nodes to be explored we will have to traverse through all the edges in the graph from each of the vertices. The graph was represented as a adjacency list due to space limitation (maximum number of nodes = 3500) .

Pseudo Code :

```
Set all nodes to "not visited";
   q = new Queue();
   q.enqueue(initial node);
   while ( q ≠ empty ) do
   {
      x = q.dequeue();
      if ( x has not been visited )
      {
         visited[x] = true;
         for (every edge (x, y))
            if ( y has not been visited )
             q.enqueue(y);
      }
   }
```