Name: Ronit Khalate.

Roll No. 2231028

Assignment No.02

Title: Implement A star Algorithm for any game search problem.

Input :

```
graph=[['A','B',1,3],
        ['A','C',2,4],
        ['A','H',7,0],
        ['B','D',4,2],
        ['B','E',6,6],
        ['C','F',3,3],
        ['C','G',2,1],
        ['D','E',7,6],
        ['D','H',5,0],
        ['F','H',1,0],
        ['G','H',2,0]]

temp=[]
temp1 =[]
for i in graph:
    temp.append(i[0])
    temp1.append(i[1])
nodes = set(temp).union (set(temp1))
def A_star(graph, costs, open, closed, cur_node):
   if cur_node in open:
      open.remove(cur_node)
   closed.add(cur_node)
   for i in graph:
      if(i[0] == cur_node and costs[i[0]]+i[2]+i[3] < costs[i[1]]):
          open.add(i[1])
          costs[i[1]]=costs[i[0]]+i[2]+i[3]
          path[i[1]] = path[i[0]] +'->' + i[1]
   costs [cur_node] = 999999
   small =min(costs, key=costs.get)
   if small not in closed:
       A_star (graph, costs, open, closed, small)
costs = dict()
temp_cost = dict()
path = dict()
for i in nodes:
   costs[i]=999999
   path[i]=' '
open = set()
closed = set()
start_node = input("Enter the Start Node:")
```

```
open.add(start_node)
path[start_node] = start_node
costs[start_node] = 0

A_star(graph, costs, open, closed, start_node)
goal_node = input("Enter the Goal Node: ")
print("Path with least cost is: ", path[goal_node])
```

Output:

```
Enter the Start Node:A

Enter the Goal Node: H

Path with least cost is:  A->C->G->H
```