LabActivity1:

**class** Node:  
 *#state=state* **def** \_\_init\_\_(self,state,parent,actions,totalCost):  
 self.state=state  
 self.parent=parent  
 self.actions=actions  
 self.totalCost=totalCost  
graph={**'A'**:Node(**'A'**,None,[**'B'**,**'C'**,**'E'**],None),  
 **'B'**:Node(**'B'**,None,[**'A'**,**'D'**,**'E'**],None),  
 **'C'**:Node(**'C'**,None,[**'A'**,**'F'**,**'G'**],None),  
 **'D'**:Node(**'D'**,None,[**'B'**,**'E'**],None),  
 **'E'**:Node(**'E'**,None,[**'A'**,**'B'**,**'D'**],None),  
 **'F'**:Node(**'F'**,None,[**'C'**],None),  
 **'G'**:Node(**'G'**,None,[**'C'**],None)}  
**print** (graph.keys())

LabActivity2:

**class** Node:  
 *#state=state* **def** \_\_init\_\_(self,state,parent,actions,totalCost):  
 self.state=state  
 self.parent=parent  
 self.actions=actions  
 self.totalCost=totalCost  
**def** BFS():  
 initialState=**'D'** goalState=**'F'** graph={**'A'**:Node(**'A'**,None,[**'B'**,**'C'**,**'E'**],None),  
 **'B'**:Node(**'B'**,None,[**'A'**,**'D'**,**'E'**],None),  
 **'C'**:Node(**'C'**,None,[**'A'**,**'F'**,**'G'**],None),  
 **'D'**:Node(**'D'**,None,[**'B'**,**'E'**],None),  
 **'E'**:Node(**'E'**,None,[**'A'**,**'B'**,**'D'**],None),  
 **'F'**:Node(**'F'**,None,[**'C'**],None),  
 **'G'**:Node(**'G'**,None,[**'C'**],None)}  
 frontier=[initialState]  
 explored=[]  
 **while** len(frontier)!=0:  
 currentNode=frontier.pop(0)  
 explored.append(currentNode)  
 **for** child **in** graph[currentNode].actions:  
 **if** child **not in** frontier **and** child **not in** explored:  
 graph[child].parent=currentNode  
 **if** graph[child].state==goalState:   
 **return** actionSequence(graph,initialState,goalState)  
 frontier.append(child)  
   
**def** actionSequence(graph,initialState,goalState):  
 solution=[goalState]  
 currentParent=graph[goalState].parent  
 **while** currentParent!=None:  
 solution.append(currentParent)  
 currentParent=graph[currentParent].parent  
 solution.reverse()  
 **return** solution  
solution=BFS()  
**print**(solution)

LabTask1:

**class** Node:  
 *#state=state* **def** \_\_init\_\_(self,state,parent,actions,totalCost):  
 self.state=state  
 self.parent=parent  
 self.actions=actions  
 self.totalCost=totalCost  
**def** BFS():  
 initialState=**'Arad'** goalState=**'Bucharest'** graph={**'Arad'**:Node(**'Arad'**,None,[**'Sibiu'**],None),  
   
 **'Sibiu'**:Node(**'Sibiu'**,None,[**'Arad'**,**'Ricuimn vilcea'**,**'Fagaras'**],None),  
 **'Fagaras'**:Node(**'Fagaras'**,None,[**'Sibiu'**,**'Bucharest'**],None),  
 **'Lugo'**:Node(**'Lugo'**,None,[**'Mehadia'**],None),  
 **'Ricuimn vilcea'**:Node(**'Ricuimn vilcea'**,None,[**'Sibiu'**,**'Pitesti'**,],None),  
 **'Mehadia'**:Node(**'Mehadia'**,None,[**'Lugo'**,**'Drobeta'**],None),  
 **'Pitesti'**:Node(**'Pitesti'**,None,[**'Bucharest'**,**'Ricuimn vilcea'**],None),  
 **'Bucharest'**:Node(**'Bucharest'**,None,[**'Gilurgiu'**,**'Pitesti'**,**'Fagaras'**],None)}   
   
 frontier=[initialState]  
 explored=[]  
 **while** len(frontier)!=0:  
 currentNode=frontier.pop(0)  
 explored.append(currentNode)  
 **for** child **in** graph[currentNode].actions:  
 **if** child **not in** frontier **and** child **not in** explored:  
 graph[child].parent=currentNode  
 **if** graph[child].state==goalState:   
 **return** actionSequence(graph,initialState,goalState)  
 frontier.append(child)  
   
**def** actionSequence(graph,initialState,goalState):  
 solution=[goalState]  
 currentParent=graph[goalState].parent  
 **while** currentParent!=None:  
 solution.append(currentParent)  
 currentParent=graph[currentParent].parent  
 solution.reverse()  
 **return** solution  
solution=BFS()  
**print**(solution)