Program: -

```
import java.util.*;
class Graph
{
   private int V;
   private LinkedList<Integer> adj[];
   public Graph(int v)
   {
      V=v;
      adj = new LinkedList[v];
      for (int i = 0; i < v; ++i)
      {
        adj[i] = new LinkedList();
          }
   }
   void addEdge(int v, int w)
   {
      adj[v].add(w);
   }
   void DFSUtil(int vertex, boolean visit[])
   {
      visit[vertex] = true;
      System.out.print(vertex + " ");
      int a = 0;
      for (int i = 0; i < adj[vertex].size(); i++)
         {
            a = adj[vertex].get(i);
            if (!visit[a])
            {
               DFSUtil(a, visit);
            }
         }
```

```
    }
    void DFS(int v)
    {
        boolean already[] = new boolean[V];
        DFSUtil(v, already);
    }
    void BFS(int s) {
        boolean visited[] = new boolean[V];
        BFSUtil(s, visited);
    }
    void BFSUtil(int s, boolean visited[]) {
        LinkedList<Integer> queue = new LinkedList<Integer>();
        visited[s] = true;
        queue.add(s);
        while (queue.size() != 0) {
            s = queue.poll();
            System.out.print(s + " ");
            Iterator<Integer> i = adj[s].listIterator();
            while (i.hasNext()) {
                int n = i.next();
                if (!visited[n]) {
                    visited[n] = true;
                    queue.add(n);
                }
            }
        }
    }
}
class Main
{
    public static void main(String[] args)
    {
```

```
        Scanner in= new Scanner(System.in);
        System.out.print("Enter the size of the graph: ");
        int n = in.nextInt();
        System.out.print("Enter the size of input: ");
        int size=in.nextInt();
        Graph g=new Graph(n);
        for(int i=0;i<size;i++){
           System.out.print("Enter edges  "+(i+1)+ " of graph: ");
           int j=in.nextInt();
           int k=in.nextInt();
           if(j<n && k<n){
             g.addEdge(j, k);
           }
           else{
              System.out.println("Invalid Input");
           }
        }
        System.out.println("Enter the starting vertex: ");
        int start=in.nextInt();
        System.out.println("DFS of Graph");
        g.DFS(start);
        System.out.println();
        System.out.println("BFS of Graph");
        g.BFS(start);
    }
}
```

Output: -

```
Enter the size of the graph: 5
Enter the size of input: 6
Enter edges  1 of graph: 0 1
Enter edges  2 of graph: 0 2
Enter edges  3 of graph: 0 3
Enter edges  4 of graph: 1 2
Enter edges  5 of graph: 2 4
Enter edges  6 of graph: 3 4
Enter the starting vertex:
0
DFS of Graph
0 1 2 4 3
BFS of Graph
0 1 2 3 4
```

Program: -

```
import java.util.*;
class Node implements Comparable<Node>{
   int vertex;
   int fScore;
   Node(int vertex, int fScore) {
      this.vertex = vertex;
      this.fScore = fScore;
   }
   public int compareTo(Node other) {
      return Integer.compare(this.fScore, other.fScore);
   }
}
public class AStarGraph {
   private int V;
   private LinkedList<Edge>[] adj;
   private int[] h;

   class Edge {
      int dest;
      int weight;
      Edge(int dest, int weight) {
         this.dest = dest;
         this.weight = weight;
      }
   }
   AStarGraph(int v) {
      V = v;
      adj = new LinkedList[v];
      for (int i=0; i<v; ++i)
         adj[i] = new LinkedList();
      h = new int[v];
```

```
}
void addEdge(int u, int v, int weight) {
    adj[u].add(new Edge(v, weight));
}
void setHeuristic(int[] heuristic) {
    h = heuristic;
}
void AStar(int start, int end) {
    PriorityQueue<Node> pq = new PriorityQueue<Node>();
    pq.add(new Node(start, h[start]));
    boolean[] visited = new boolean[V];
    int[] gScore = new int[V];
    Arrays.fill(gScore, Integer.MAX_VALUE);
    gScore[start] = 0;
    System.out.print("The Shortest Path: ");
    while (!pq.isEmpty()) {
        Node curr = pq.poll();
        int u = curr.vertex;
        System.out.print(u+" ");
        if (u == end) {
            System.out.println("\nShortest path from " + start + " to " + end + " has cost of: " + gScore[u]);
            return;
        }
        visited[u] = true;
        for (Edge e : adj[u]) {
            int v = e.dest;
            int w = e.weight;
            if (!visited[v]) {
                int fScore = gScore[u] + w + h[v];
                if (fScore < gScore[v]) {
                    gScore[v] = fScore;
                    pq.add(new Node(v, fScore));}}}}
```

```
    System.out.println("No path found from " + start + " to " + end);}
  public static void main(String args[]) {
    Scanner in= new Scanner(System.in);
    System.out.print("Enter the size of the graph: ");
    int n = in.nextInt();
    AStarGraph g = new AStarGraph(n);
    System.out.print("Enter the size of input: ");
    int size=in.nextInt();
    System.out.println("Enter edges of graph ");
    for(int i=0;i<size;i++){
      System.out.print("Enter "+(i+1)+ " edges and weight of that edges: ");
      int j=in.nextInt();
      int k=in.nextInt();
      int w=in.nextInt();
      if(j<n && k<n){
        g.addEdge(j, k,w);
      }
      else{
        System.out.println("Invalid Input");
      }}
    int[] heuristic = new int[n];
    System.out.println("Enter heuristic of the edges of graph ");
    for(int i=0;i<n;i++){
      System.out.print("Enter "+(i+1)+ " edges heuristic value: ");
      heuristic[i]=in.nextInt();
    }
    g.setHeuristic(heuristic);
    System.out.print("Enter the starting and the ending vertex where you want to find the shortest
distance: ");
    int start=in.nextInt();
    int end=in.nextInt();
    g.AStar(start, end);
  }}
```

Output: -

Enter the size of the graph: 5

Enter the size of input: 7

Enter edges of graph

Enter 1 edges and weight of that edges: 0 1 4

Enter 2 edges and weight of that edges: 0 2 2

Enter 3 edges and weight of that edges: 1 2 1

Enter 4 edges and weight of that edges: 1 3 5

Enter 5 edges and weight of that edges: 2 3 8

Enter 6 edges and weight of that edges: 2 4 10

Enter 7 edges and weight of that edges: 3 4 2

Enter heuristic of the edges of graph

Enter 1 edges heuristic value: 7

Enter 2 edges heuristic value: 6

Enter 3 edges heuristic value: 2

Enter 4 edges heuristic value: 1

Enter 5 edges heuristic value: 0

Enter the starting and the ending vertex where you want to find the shortest distance: 0 4

The Shortest Path: 0 2 1 3 4

Shortest path from 0 to 4 has cost of: 14

Program: -

```
import java.util.*;
public class NQueensProblem {
   private int[] queens;
   private int numSolutions;
   public NQueensProblem(int n) {
      queens = new int[n];
   }
   public void solve() {
      solve(0);
   }
   private void solve(int row) {
      if (row == queens.length) {
         numSolutions++;
         printSolution();
      } else {
         for (int col = 0; col < queens.length; col++) {
            queens[row] = col;
            if (isValid(row, col)) {
               solve(row + 1);
            }}}}
   private boolean isValid(int row, int col) {
      for (int i = 0; i < row; i++) {
         int diff = Math.abs(queens[i] - col);
         if (diff == 0 || diff == row - i) {
            return false;
         }
      }
      return true;
   }
   private void printSolution() {
      if(numSolutions==1){
```

```
        System.out.print("Solution: ");
    for (int i = 0; i < queens.length; i++) {
        System.out.print(queens[i] + " ");
    }
    System.out.println();
    System.out.println("The Matrix Representation:");
    int [][]arr=new int[queens.length][queens.length];
    for(int i=0;i< queens.length;i++){
        for(int j=0;j<queens.length;j++){
            if((j)==queens[i]){
                arr[i][j]=1;
            }
            else{
                arr[i][j]=0;
            }
        }
    }
    for(int i=0;i< queens.length;i++){
        for(int j=0;j<queens.length;j++){
            System.out.print(arr[i][j]+" ");
        }
        System.out.println();
    }
    }
}
public static void main(String[] args) {
    Scanner in= new Scanner(System.in);
System.out.print("Enter N Queens Problem: ");
int n = in.nextInt();
    NQueensProblem NQueensProblem = new NQueensProblem(n);
    NQueensProblem.solve();
}}
```

Output: -

Enter N Queens Problem: 4

Solution: 1 3 0 2

The Matrix Representation:

0 1 0 0

0 0 0 1

1 0 0 0

Enter N Queens Problem: 6

Solution: 1 3 5 0 2 4

The Matrix Representation:

0 1 0 0 0 0

0 0 0 1 0 0

0 0 0 0 0 1

1 0 0 0 0 0

0 0 1 0 0 0

0 0 0 0 1 0

Enter N Queens Problem: 8

Solution: 0 4 7 5 2 6 1 3

The Matrix Representation:

1 0 0 0 0 0 0 0

0 0 0 0 1 0 0 0

0 0 0 0 0 0 0 1

0 0 0 0 0 1 0 0

0 0 1 0 0 0 0 0

0 0 0 0 0 0 1 0

0 1 0 0 0 0 0 0

0 0 0 1 0 0 0 0

Program: -

```
import java.util.*;
public class SelectionSort {
   public static void selectionSort(int[] arr) {
      int n = arr.length;
      for (int i = 0; i < n-1; i++) {
         int min_idx = i;
         for (int j = i+1; j < n; j++) {
            if (arr[j] < arr[min_idx]) {
               min_idx = j;
            }}
         int temp = arr[min_idx];
         arr[min_idx] = arr[i];
         arr[i] = temp;
      }}
   public static void main(String args[]) {
      Scanner in= new Scanner(System.in);
      System.out.print("Enter the size of the graph: ");
      int n = in.nextInt();
      int arr[] = new int[n];
      System.out.println("Enter the elements of the array");
      for (int i = 0; i < n; i++) {
         System.out.print("Enter the "+(i+1)+" element: ");
         arr[i]=in.nextInt();}
      System.out.println("Unsorted array:");
      for (int i = 0; i <n; i++) {
         System.out.print(arr[i] + " ");}
      selectionSort(arr);
      System.out.println("\nSorted array:");
      for (int i = 0; i <n; i++) {
         System.out.print(arr[i] + " ");
}}}
```

Output: -

Enter the size of the input: 5

Enter the elements of the array

Enter the 1 element: 64

Enter the 2 element: 25

Enter the 3 element: 12

Enter the 4 element: 23

Enter the 5 element: 16

Unsorted array:

64 25 12 23 16

Sorted array:

12 16 23 25 64

Program: -

```
import java.util.*;
class Job {
   int id, deadline, profit;
   public Job(int id, int deadline, int profit) {
      this.id = id;
      this.deadline = deadline;
      this.profit = profit;
   }
}
class JobScheduling {
   public static void main(String[] args) {
      Scanner in = new Scanner(System.in);
      System.out.print("Enter the no of Job you want to enter: ");
      int n=in.nextInt();
      Job[] jobs =new Job[n];
      System.out.print("Enter the details of the Job: \n");
      for(int i=0;i<n;i++){
         System.out.println("Job "+(i+1)+" :");
         System.out.print("Enter the id of Job: ");
         int id=in.nextInt();
         System.out.print("Enter the deadline of Job: ");
         int deadline=in.nextInt();
         System.out.print("Enter the profit of Job: ");
         int profit=in.nextInt();
         jobs[i]=new Job(id, deadline, profit);
      }
      Arrays.sort(jobs, (a, b) -> b.profit - a.profit);
      int maxDeadline = Integer.MIN_VALUE;
      for (Job job : jobs) {
         maxDeadline = Math.max(maxDeadline, job.deadline);
      }
```

```
        int[] slots = new int[maxDeadline + 1];

        int totalProfit = 0;
        for (Job job : jobs) {

            for (int i = job.deadline; i > 0; i--) {
                if (slots[i] == 0) {
                    slots[i] = job.id;
                    totalProfit += job.profit;
                    break;
                }
            }
        }

        System.out.print("Scheduled Jobs: ");
        for (int i = 1; i < slots.length; i++) {
            if (slots[i] != 0) {
                System.out.print(slots[i] + " ");
            }
        }
        System.out.println("\nTotal Profit: " + totalProfit);
    }
}
```

Output: -

Enter the no of Job you want to enter: 5

Enter the details of the Job:

Job 1 :

Enter the id of Job: 1

Enter the deadline of Job: 2

Enter the profit of Job: 100

Job 2 :

Enter the id of Job: 2

Enter the deadline of Job: 1

Enter the profit of Job: 19

Job 3 :

Enter the id of Job: 3

Enter the deadline of Job: 2

Enter the profit of Job: 27

Job 4 :

Enter the id of Job: 4

Enter the deadline of Job: 1

Enter the profit of Job: 25

Job 5 :

Enter the id of Job: 5

Enter the deadline of Job: 3

Enter the profit of Job: 15

Scheduled Jobs: 3 1 5

Total Profit: 142

Program: -

```
import java.util.*;
public class PrimMST {
   public static void prim(int[][] graph, int numVertices) {
      int[] parent = new int[numVertices];
      int[] key = new int[numVertices];
      boolean[] mstSet = new boolean[numVertices];
      for (int i = 0; i < numVertices; i++) {
         key[i] = Integer.MAX_VALUE;
         mstSet[i] = false;
      }
      key[0] = 0;
      parent[0] = -1;
      for (int count = 0; count < numVertices - 1; count++) {
         int u = minKey(key, mstSet);
         mstSet[u] = true;
         for (int v = 0; v < numVertices; v++) {
            if (graph[u][v] != 0 && mstSet[v] == false && graph[u][v] < key[v]) {
               parent[v] = u;
               key[v] = graph[u][v];
            }
         }
      }
      int sum=0;
      for (int i = 0; i < numVertices; i++) {
         sum += key[i];
      }
      printMST(parent, graph,sum);
   }
   public static int minKey(int[] key, boolean[] mstSet) {
      int min = Integer.MAX_VALUE, minIndex = -1;
```

```
        for (int i = 0; i < key.length; i++) {
            if (mstSet[i] == false && key[i] < min) {
                min = key[i];
                minIndex = i;
            }
        }
        return minIndex;
    }
    public static void printMST(int[] parent, int[][] graph,int sum) {
        System.out.println("Edge \tWeight");
        for (int i = 1; i < parent.length; i++) {
            System.out.println(parent[i] + " - " + i + "\t" + graph[i][parent[i]]);
        }
        System.out.println("Minimum weight of MST: " + sum);
    }
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.print("Enter the  size of the graph: ");
        int n = in.nextInt();
        int[][] graph = new int[n][n];
        for(int i=0;i<n;i++){
            for(int j=0;j<n;j++){
                System.out.print("Enter the weight "+i+ "-> "+j+" of the graph: ");
                graph[i][j]=in.nextInt();
            }
        }
        prim(graph, n);
    }

}
```

Output: -

```
Enter the  size of the graph: 5
Enter the weight 0-> 0 of the graph: 0
Enter the weight 0-> 1 of the graph: 2
Enter the weight 0-> 2 of the graph: 0
Enter the weight 0-> 3 of the graph: 6
Enter the weight 0-> 4 of the graph: 0
Enter the weight 1-> 0 of the graph: 2
Enter the weight 1-> 1 of the graph: 0
Enter the weight 1-> 2 of the graph: 3
Enter the weight 1-> 3 of the graph: 8
Enter the weight 1-> 4 of the graph: 5
Enter the weight 2-> 0 of the graph: 0
Enter the weight 2-> 1 of the graph: 3
Enter the weight 2-> 2 of the graph: 0
Enter the weight 2-> 3 of the graph: 0
Enter the weight 2-> 4 of the graph: 7
Enter the weight 3-> 0 of the graph: 6
Enter the weight 3-> 1 of the graph: 8
Enter the weight 3-> 2 of the graph: 0
Enter the weight 3-> 3 of the graph: 0
Enter the weight 3-> 4 of the graph: 9
Enter the weight 4-> 0 of the graph: 0
Enter the weight 4-> 1 of the graph: 5
Enter the weight 4-> 2 of the graph: 7
Enter the weight 4-> 3 of the graph: 9
Enter the weight 4-> 4 of the graph: 0
Edge    Weight
0 - 1   2
1 - 2   3
0 - 3   6
1 - 4   5
Minimum weight of MST: 16
```

Program: -

```
import java.util.*;
public class KruskalMST {
   class Edge implements Comparable<Edge> {
      int src, dest, weight;
      public int compareTo(Edge other) {
         return this.weight - other.weight;
      }
   }
   public void kruskal(int[][] graph, int numVertices) {
      List<Edge> edges = new ArrayList<>();
      for (int i = 0; i < numVertices; i++) {
         for (int j = i + 1; j < numVertices; j++) {
            if (graph[i][j] != 0) {
               Edge edge = new Edge();
               edge.src = i;
               edge.dest = j;
               edge.weight = graph[i][j];
               edges.add(edge);
            }
         }
      }
      Collections.sort(edges);
      int[] parent = new int[numVertices];
      for (int i = 0; i < numVertices; i++) {
         parent[i] = i;
      }
      List<Edge> mst = new ArrayList<>();
      for (Edge edge : edges) {
         int srcParent = find(parent, edge.src);
         int destParent = find(parent, edge.dest);
         if (srcParent != destParent) {
```

```
            mst.add(edge);
            parent[srcParent] = destParent;
        }
    }
    int sum=0;
    System.out.println("Edges in the MST:");
    for (Edge edge : mst) {
        System.out.println(edge.src + " - " + edge.dest + " : " + edge.weight);
        sum+=edge.weight;
    }
    System.out.println("Minimum weight of MST: " + sum);
}
private int find(int[] parent, int i) {
    if (parent[i] != i) {
        parent[i] = find(parent, parent[i]);
    }
    return parent[i];
}

public static void main(String[] args) {
    Scanner in = new Scanner(System.in);
    System.out.print("Enter the  size of the graph: ");
    int n = in.nextInt();
    int[][] graph = new int[n][n];
    for(int i=0;i<n;i++){
        for(int j=0;j<n;j++){
            System.out.print("Enter the weight "+i+ "-> "+j+" of the graph: ");
            graph[i][j]=in.nextInt();
        }
    }
    KruskalMST kruskal = new KruskalMST();
    kruskal.kruskal(graph, n);}}
```

Output: -

Enter the size of the graph: 5

Enter the weight 0-> 0 of the graph: 0

Enter the weight 0-> 1 of the graph: 2

Enter the weight 0-> 2 of the graph: 0

Enter the weight 0-> 3 of the graph: 6

Enter the weight 0-> 4 of the graph: 0

Enter the weight 1-> 0 of the graph: 2

Enter the weight 1-> 1 of the graph: 0

Enter the weight 1-> 2 of the graph: 3

Enter the weight 1-> 3 of the graph: 8

Enter the weight 1-> 4 of the graph: 5

Enter the weight 2-> 0 of the graph: 0

Enter the weight 2-> 1 of the graph: 3

Enter the weight 2-> 2 of the graph: 0

Enter the weight 2-> 3 of the graph: 0

Enter the weight 2-> 4 of the graph: 7

Enter the weight 3-> 0 of the graph: 6

Enter the weight 3-> 1 of the graph: 8

Enter the weight 3-> 2 of the graph: 0

Enter the weight 3-> 3 of the graph: 0

Enter the weight 3-> 4 of the graph: 9

Enter the weight 4-> 0 of the graph: 0

Enter the weight 4-> 1 of the graph: 5

Enter the weight 4-> 2 of the graph: 7

Enter the weight 4-> 3 of the graph: 9

Enter the weight 4-> 4 of the graph: 0

Edges in the MST:
0 - 1 : 2
1 - 2 : 3
1 - 4 : 5
0 - 3 : 6
Minimum weight of MST: 16

Program: -

```
import java.util.*;
public class DijkstraMST {
   private int numVertices;
   private int[] dist;
   private boolean[] visited;
   private int[][] graph;
   public DijkstraMST(int[][] graph, int numVertices) {
      this.numVertices = numVertices;
      this.graph = graph;
      this.dist = new int[numVertices];
      this.visited = new boolean[numVertices];
   }
   public void dijkstra(int startVertex) {
      for (int i = 0; i < numVertices; i++) {
         dist[i] = Integer.MAX_VALUE;
         visited[i] = false;
      }
      dist[startVertex] = 0;
      for (int i = 0; i < numVertices - 1; i++) {
         int u = minDistance(dist, visited);
         visited[u] = true;
         for (int v = 0; v < numVertices; v++) {
            if (!visited[v] && graph[u][v] != 0 && dist[u] != Integer.MAX_VALUE && dist[u] + graph[u][v] <
dist[v]) {
               dist[v] = dist[u] + graph[u][v];
            }}}
      printMST(startVertex);
   }

   private int minDistance(int[] dist, boolean[] visited) {
      int minDist = Integer.MAX_VALUE;
```

```
    int minIndex = -1;
    for (int i = 0; i < numVertices; i++) {
        if (!visited[i] && dist[i] <= minDist) {
            minDist = dist[i];
            minIndex = i;
        }
    }
    return minIndex;
}

private void printMST(int startVertex) {
    System.out.println("Vertex \t Distance from Source");
    for (int i = 0; i < numVertices; i++) {
        System.out.println(i + "\t" + dist[i]);
    }
}

public static void main(String[] args) {
    Scanner in = new Scanner(System.in);
    System.out.print("Enter the  size of the graph: ");
    int n = in.nextInt();
    int[][] graph = new int[n][n];
    for(int i=0;i<n;i++){
        for(int j=0;j<n;j++){
            System.out.print("Enter the weight "+i+ "-> "+j+" of the graph: ");
            graph[i][j]=in.nextInt();
        }}
    DijkstraMST dijkstra = new DijkstraMST(graph,n);
    System.out.print("Enter the starting vertex of the graph: ");
    int vertex=in.nextInt();
    dijkstra.dijkstra(vertex);
}}
```

Output: -

```
Enter the  size of the graph: 5
Enter the weight 0-> 0 of the graph: 0
Enter the weight 0-> 1 of the graph: 2
Enter the weight 0-> 2 of the graph: 0
Enter the weight 0-> 3 of the graph: 6
Enter the weight 0-> 4 of the graph: 0
Enter the weight 1-> 0 of the graph: 2
Enter the weight 1-> 1 of the graph: 0
Enter the weight 1-> 2 of the graph: 3
Enter the weight 1-> 3 of the graph: 8
Enter the weight 1-> 4 of the graph: 5
Enter the weight 2-> 0 of the graph: 0
Enter the weight 2-> 1 of the graph: 3
Enter the weight 2-> 2 of the graph: 0
Enter the weight 2-> 3 of the graph: 0
Enter the weight 2-> 4 of the graph: 7
Enter the weight 3-> 0 of the graph: 6
Enter the weight 3-> 1 of the graph: 8
Enter the weight 3-> 2 of the graph: 0
Enter the weight 3-> 3 of the graph: 0
Enter the weight 3-> 4 of the graph: 9
Enter the weight 4-> 0 of the graph: 0
Enter the weight 4-> 1 of the graph: 5
Enter the weight 4-> 2 of the graph: 7
Enter the weight 4-> 3 of the graph: 9
Enter the weight 4-> 4 of the graph: 0
Enter the starting vertex of the graph: 1
Vertex   Distance from Source
0      2
1      0
2      3
3      8
4      5
```

Program: -

```
import random

responses = {
    "hi": "Hello, welcome to Enterprise Bot! How can I assist you today?",
    "services": "We offer the following services:\n- IT Support\n- Software Development\n- Cloud Computing\n- Data Analytics\nWhich service are you interested in?",
    "it support": "Great, let me transfer you to our IT support team.",
    "software development": "Great, let me transfer you to our software development team.",
    "cloud computing": "Great, let me transfer you to our cloud computing team.",
    "data analytics": "Great, let me transfer you to our data analytics team.",
    "default": "I'm sorry, I didn't understand. Can you please rephrase?"
}
def get_response(user_input):
    user_input = user_input.lower()
    if "it support" in user_input:
        return responses["it support"]
    elif "software development" in user_input:
        return responses["software development"]
    elif "cloud computing" in user_input:
        return responses["cloud computing"]
    elif "data analytics" in user_input:
        return responses["data analytics"]
    elif "services" in user_input:
        return responses["services"]
    elif "hi" in user_input:
        return responses["hi"]
    elif "bye" in user_input:
        return "Thank you for contacting Enterprise Bot. Have a nice day!"
    else:
        return responses["default"]
```

```
print("Hello, welcome to Enterprise Bot! How can I assist you today?")
while True:
    user_input = input("You: ")
    if "bye" in user_input:
        print(get_response(user_input))
        break
    else:
        print(get_response(user_input))
```

Output: -

Hello, welcome to Enterprise Bot! How can I assist you today?

You: hi

Hello, welcome to Enterprise Bot! How can I assist you today?

You: services

We offer the following services:

- IT Support

- Software Development

- Cloud Computing

- Data Analytics

Which service are you interested in?

You: IT support

Great, let me transfer you to our IT support team.

You: Software Development

Great, let me transfer you to our software development team.

You: Cloud Computing

Great, let me transfer you to our cloud computing team.

You: Data Analytics

Great, let me transfer you to our data analytics team.

You: bye

Thank you for contacting Enterprise Bot. Have a nice day!

Program: -

```
rules = {
   "rule1": "If the employee meets all project deadlines, add 20 points to their score.",
   "rule2": "If the employee consistently exceeds expectations, add 30 points to their score.",
   "rule3": "If the employee shows initiative and takes on additional responsibilities, add 15 points to their score.",
   "rule4": "If the employee is frequently absent or misses deadlines, subtract 25 points from their score.",
   "rule5": "If the employee consistently performs below expectations, subtract 35 points from their score."
}


def evaluate_employee_performance(deadlines_met, expectations_exceeded, initiative_taken, absences, performance_below_expectations):
   score = 0
   if deadlines_met:
      score += 20
   if expectations_exceeded:
      score += 30
   if initiative_taken:
      score += 15
   if absences:
      score -= 25
   if performance_below_expectations:
      score -= 35
   return score


employee_data={}
n=int(input("Enter the number of data of employee you want to insert: "))
for i in range(0,n):
   name=input("Enter the name of the employee: ")
   data={
        "deadlines_met":bool(int(input("Enter the performance of deadlines met in terms of 0 or 1: "))),
        "expectations_exceeded": bool(int(input("Enter the performance of expectations exceeded in terms of 0 or 1: "))),
```

```
        "initiative_taken": bool(int(input("Enter the performance of initiative taken in terms of 0 or 1: "))),
        "absences": bool(int(input("Enter the performance of absences in terms of 0 or 1: "))),
        "performance_below_expectations":bool(int(input("Enter the performance of performance below expectations in terms  0 or 1: ")))
        }
    employee_data[name]=data
print("Rules for employee evaluation")
for rule in rules.values():
        print(f"- {rule}")
for name, data in employee_data.items():
    score = evaluate_employee_performance(data["deadlines_met"], data["expectations_exceeded"], data["initiative_taken"], data["absences"], data["performance_below_expectations"])
    print(f"Employee {name} scored {score} points")
```

Output: -

Enter the number of data of employee you want to insert: 3

Enter the name of the employee: Kushal

Enter the performance of deadlines met in terms of 0 or 1: 0

Enter the performance of expectations exceeded in terms of 0 or 1: 1

Enter the performance of initiative taken in terms of 0 or 1: 1

Enter the performance of absences in terms of 0 or 1: 0

Enter the performance of performance below expectations in terms 0 or 1: 0

Enter the name of the employee: Soham

Enter the performance of deadlines met in terms of 0 or 1: 1

Enter the performance of expectations exceeded in terms of 0 or 1: 0

Enter the performance of initiative taken in terms of 0 or 1: 1

Enter the performance of absences in terms of 0 or 1: 0

Enter the performance of performance below expectations in terms 0 or 1: 1

Enter the name of the employee: Swapnil

Enter the performance of deadlines met in terms of 0 or 1: 0

Enter the performance of expectations exceeded in terms of 0 or 1: 1

Enter the performance of initiative taken in terms of 0 or 1: 1

Enter the performance of absences in terms of 0 or 1: 1

Enter the performance of performance below expectations in terms 0 or 1: 0

Rules for employee evaluation

- If the employee meets all project deadlines, add 20 points to their score.

- If the employee consistently exceeds expectations, add 30 points to their score.

- If the employee shows initiative and takes on additional responsibilities, add 15 points to their score.

- If the employee is frequently absent or misses deadlines, subtract 25 points from their score.

- If the employee consistently performs below expectations, subtract 35 points from their score.

Employee Kushal scored 45 points

Employee Soham scored 0 points

Employee Swapnil scored 20 points