Data Structures and Algorithms

CSE2001

Assignment - 3

Yashwanth Reddy
19BCE7362
Date- 27thJuly2021

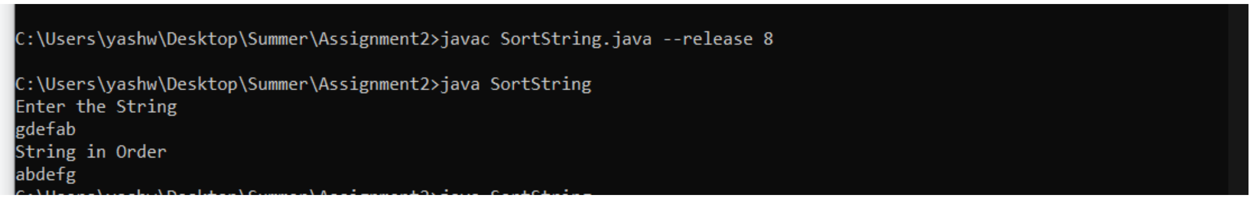
Problem-1 : Write an algorithm to read N individual characters and display them in alphabetical order using merge sort. Write down the code and execute it. Upload the code and execution results as well.

Code

```
import java.util.*;
public class SortString{
    static final int MAX_CHAR = 26;

    static void sortString(String str) {
        int letters[] = new int[MAX_CHAR];
        for (char x : str.toCharArray()) {
            letters[x - 'a']++;
        }
        for (int i = 0; i < MAX_CHAR; i++) {
            for (int j = 0; j < letters[i]; j++) {

                System.out.print((char) (i + 'a'));
            }
        }
```

```
  }

  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.println("Enter the String");
    String c = sc.nextLine();

    System.out.println("String in Order");
    sortString(c);

  }
}
```

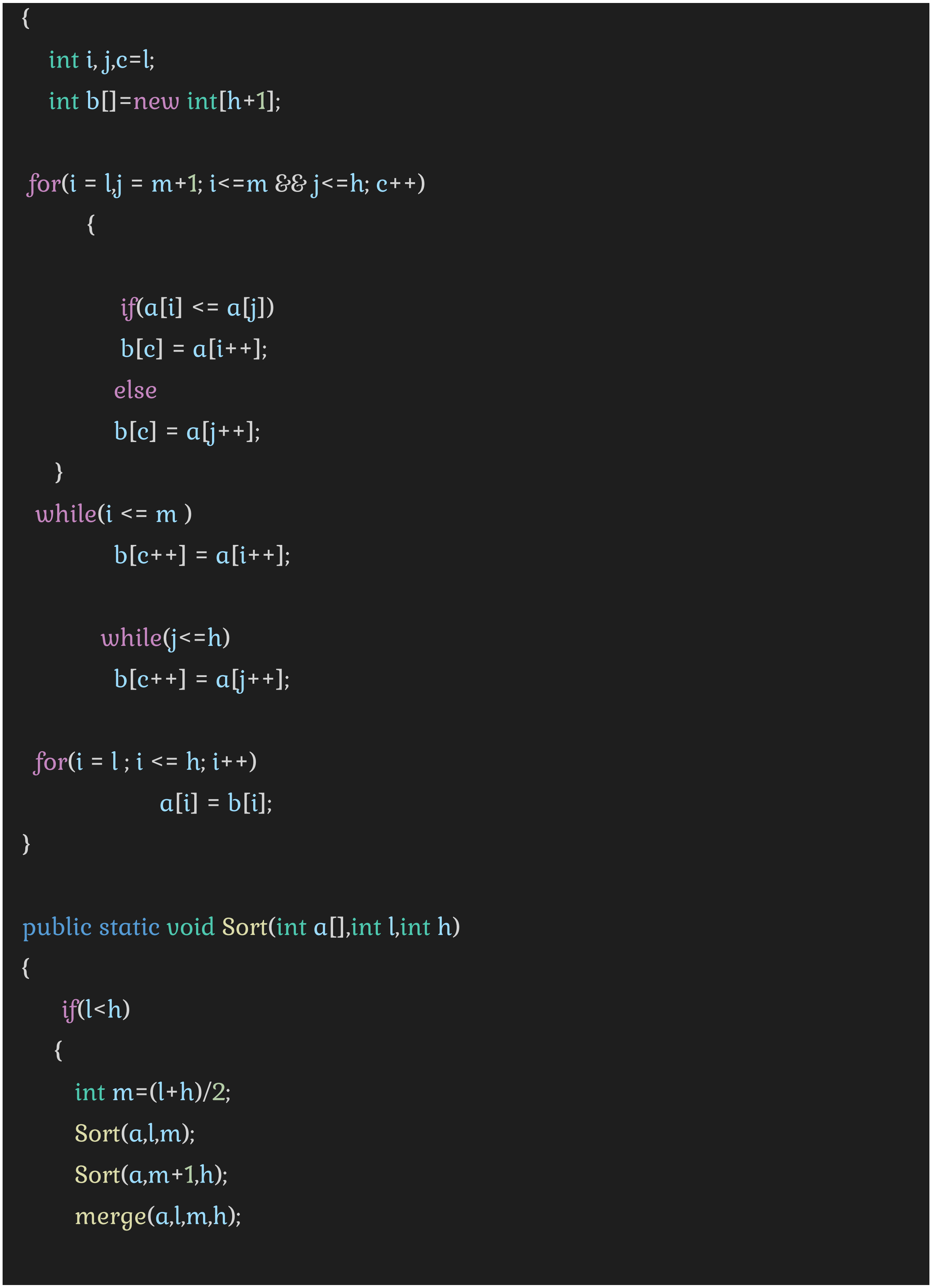
Output



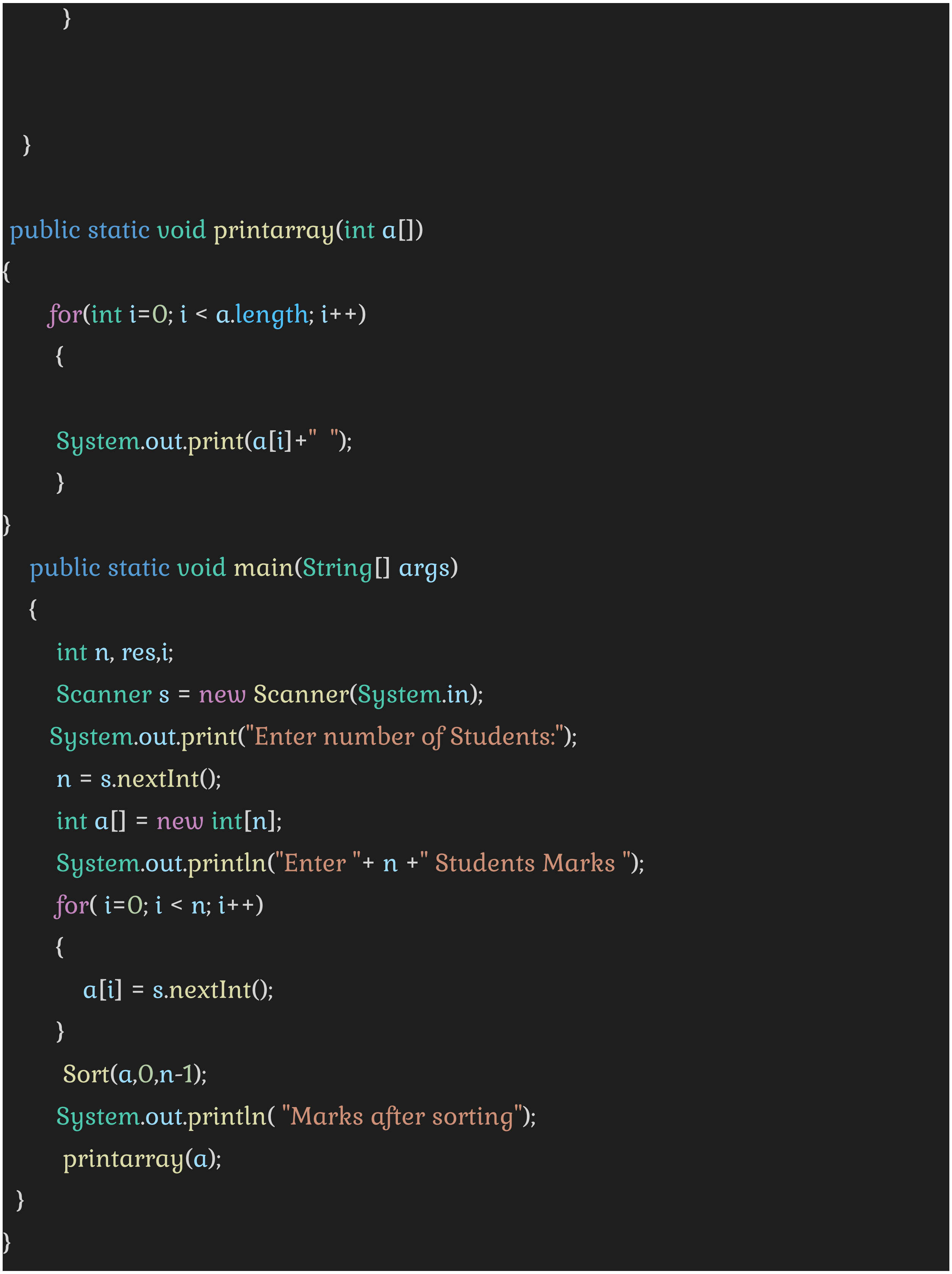
```
C:\Users\yashw\Desktop\Summer\Assignment2>javac SortString.java --release 8

C:\Users\yashw\Desktop\Summer\Assignment2>java SortString
Enter the String
gdefab
String in Order
abdefg
C:\Users\yashw\Desktop\Summer\Assignment2>java SortString
```

Code

```
import java.util.*;

public class MSort
{


 public static void merge(int a[],int l,int m,int h)
```

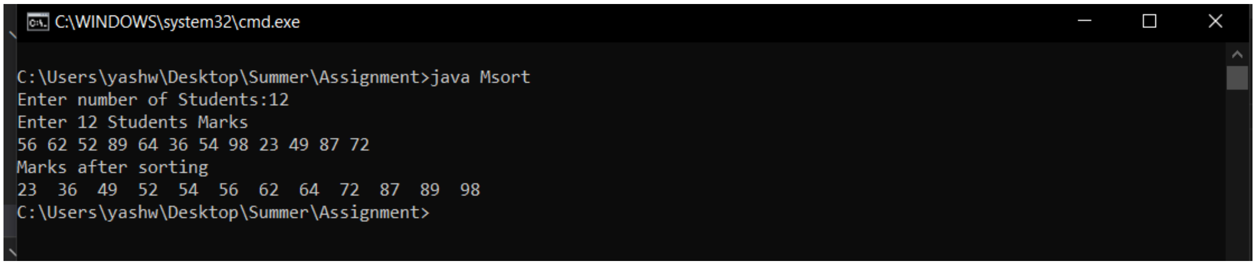


```
{
   int i, j,c=l;
   int b[]=new int[h+1];

 for(i = l,j = m+1; i<=m && j<=h; c++)
        {

            if(a[i] <= a[j])
            b[c] = a[i++];
           else
           b[c] = a[j++];
    }
  while(i <= m )
           b[c++] = a[i++];

          while(j<=h)
           b[c++] = a[j++];

  for(i = l ; i <= h; i++)
                 a[i] = b[i];
}

public static void Sort(int a[],int l,int h)
{
     if(l<h)
    {
      int m=(l+h)/2;
      Sort(a,l,m);
      Sort(a,m+1,h);
      merge(a,l,m,h);
```



```
       }



  }

 public static void printarray(int a[])
{
      for(int i=0; i < a.length; i++)
       {


       System.out.print(a[i]+"  ");
       }
}
   public static void main(String[] args)
   {
       int n, res,i;
       Scanner s = new Scanner(System.in);
      System.out.print("Enter number of Students:");
       n = s.nextInt();
       int a[] = new int[n];
       System.out.println("Enter "+ n +" Students Marks ");
       for( i=0; i < n; i++)
       {
           a[i] = s.nextInt();
       }
        Sort(a,0,n-1);
       System.out.println( "Marks after sorting");
        printarray(a);
  }
}
```

Output

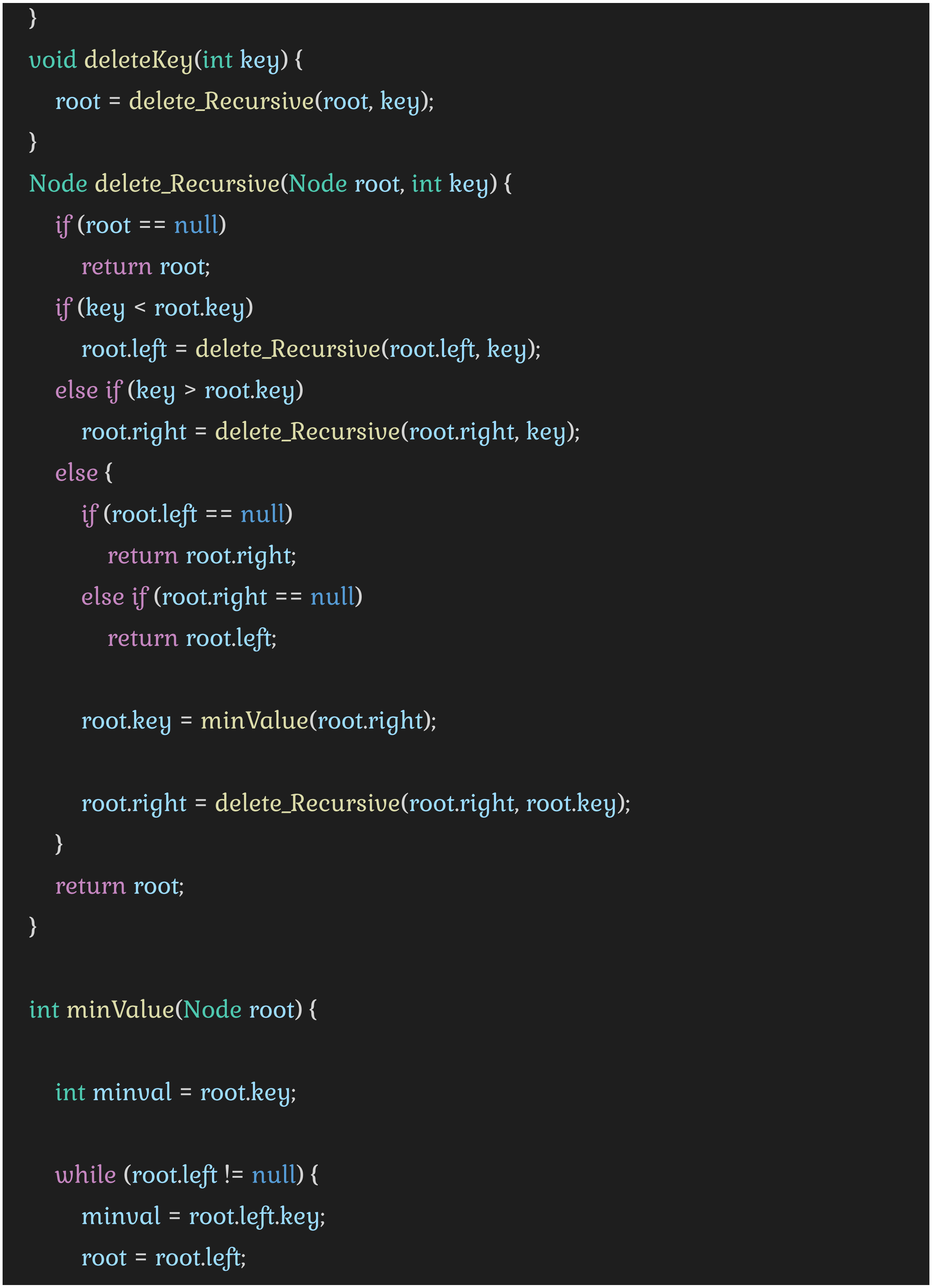


```
C:\Users\yashw\Desktop\Summer\Assignment>java Msort
Enter number of Students:12
Enter 12 Students Marks
56 62 52 89 64 36 54 98 23 49 87 72
Marks after sorting
23  36  49  52  54  56  62  64  72  87  89  98
C:\Users\yashw\Desktop\Summer\Assignment>
```

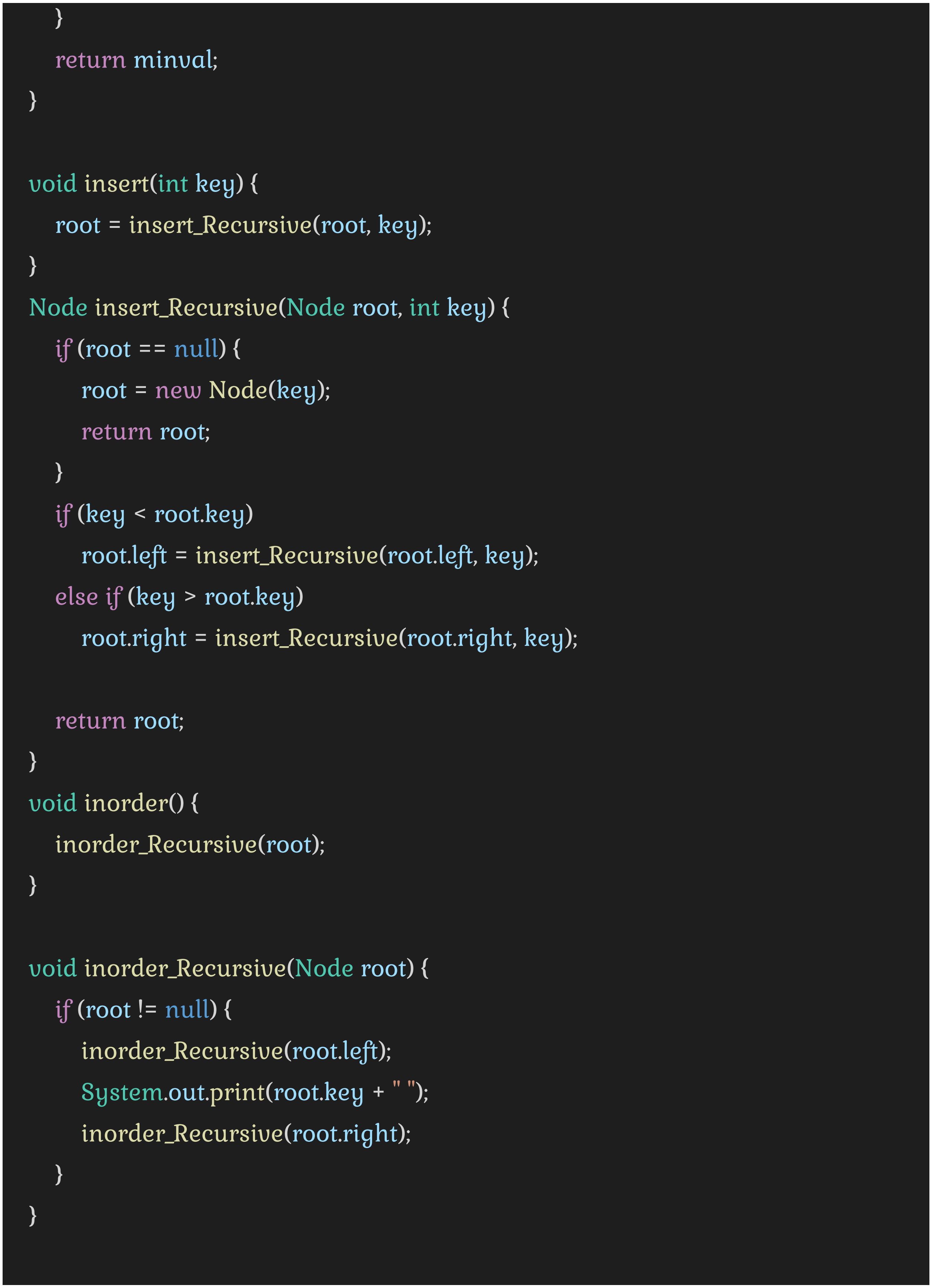
Problem-2 : Get 30 numbers from the user and store in array. Create a Binary search tree in the sequence of input. Perform the following:
(i) Insert an element into BST.
(ii) Delete an element from BST.
(iii) Search an element from BST

Code

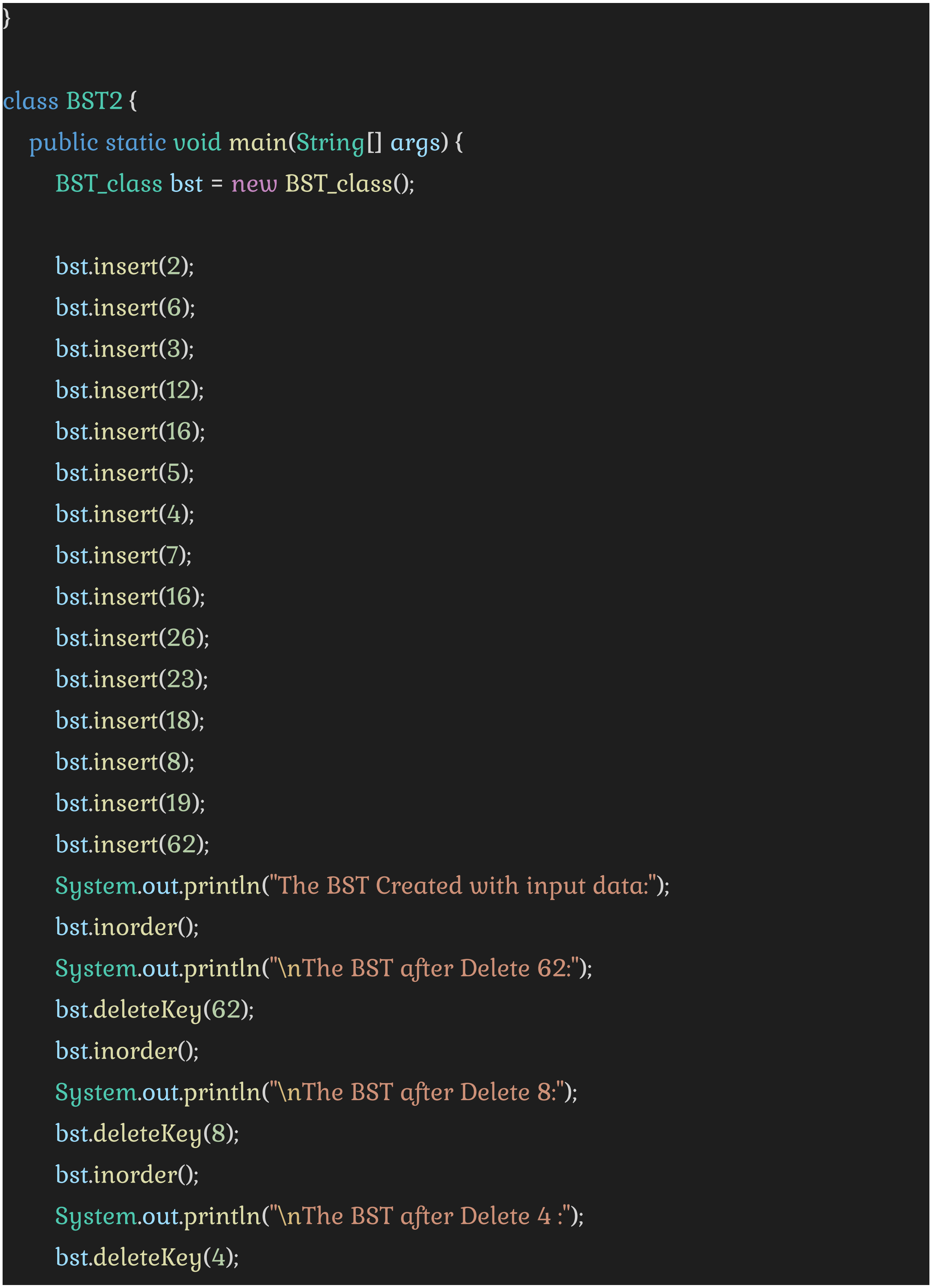
```
class BST_class {

  class Node {
    int key;
    Node left, right;

    public Node(int data) {
      key = data;
      left = right = null;
    }
  }
  Node root;
  BST_class() {
    root = null;
```



```java
}
void deleteKey(int key) {
    root = delete_Recursive(root, key);
}
Node delete_Recursive(Node root, int key) {
    if (root == null)
        return root;
    if (key < root.key)
        root.left = delete_Recursive(root.left, key);
    else if (key > root.key)
        root.right = delete_Recursive(root.right, key);
    else {
        if (root.left == null)
            return root.right;
        else if (root.right == null)
            return root.left;

        root.key = minValue(root.right);

        root.right = delete_Recursive(root.right, root.key);
    }
    return root;
}

int minValue(Node root) {

    int minval = root.key;

    while (root.left != null) {
        minval = root.left.key;
        root = root.left;
```



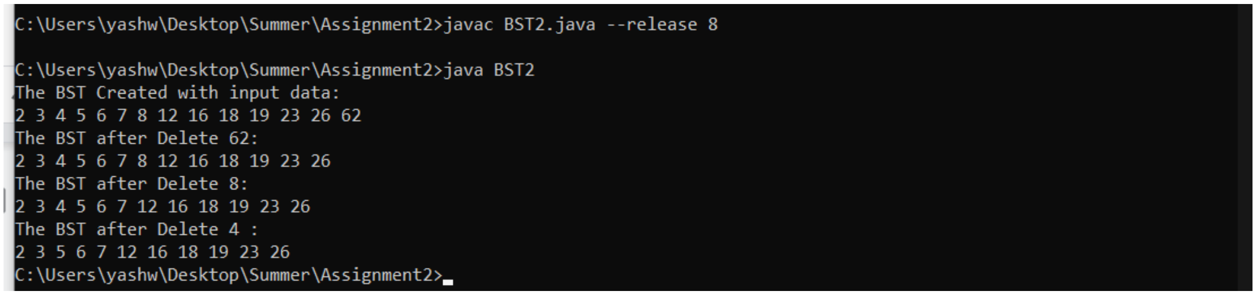
```
    }
    return minval;
}

void insert(int key) {
    root = insert_Recursive(root, key);
}
Node insert_Recursive(Node root, int key) {
    if (root == null) {
        root = new Node(key);
        return root;
    }
    if (key < root.key)
        root.left = insert_Recursive(root.left, key);
    else if (key > root.key)
        root.right = insert_Recursive(root.right, key);

    return root;
}
void inorder() {
    inorder_Recursive(root);
}

void inorder_Recursive(Node root) {
    if (root != null) {
        inorder_Recursive(root.left);
        System.out.print(root.key + " ");
        inorder_Recursive(root.right);
    }
}
```



```
}

class BST2 {
    public static void main(String[] args) {
        BST_class bst = new BST_class();

        bst.insert(2);
        bst.insert(6);
        bst.insert(3);
        bst.insert(12);
        bst.insert(16);
        bst.insert(5);
        bst.insert(4);
        bst.insert(7);
        bst.insert(16);
        bst.insert(26);
        bst.insert(23);
        bst.insert(18);
        bst.insert(8);
        bst.insert(19);
        bst.insert(62);
        System.out.println("The BST Created with input data:");
        bst.inorder();
        System.out.println("\nThe BST after Delete 62:");
        bst.deleteKey(62);
        bst.inorder();
        System.out.println("\nThe BST after Delete 8:");
        bst.deleteKey(8);
        bst.inorder();
        System.out.println("\nThe BST after Delete 4 :");
        bst.deleteKey(4);
```

```
      bst.inorder();
   }
}
```

Output

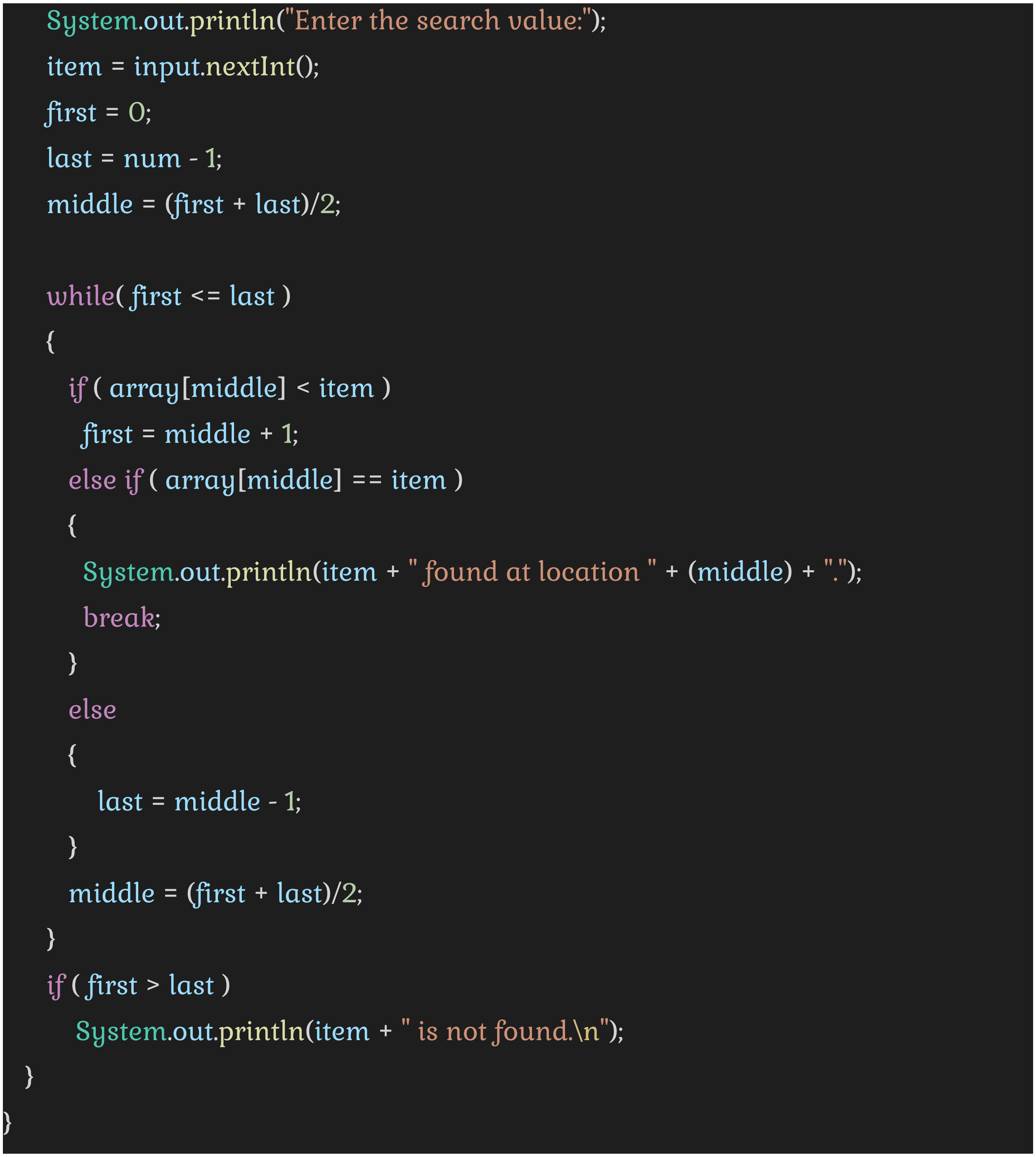


```
C:\Users\yashw\Desktop\Summer\Assignment2>javac BST2.java --release 8

C:\Users\yashw\Desktop\Summer\Assignment2>java BST2
The BST Created with input data:
2 3 4 5 6 7 8 12 16 18 19 23 26 62
The BST after Delete 62:
2 3 4 5 6 7 8 12 16 18 19 23 26
The BST after Delete 8:
2 3 4 5 6 7 12 16 18 19 23 26
The BST after Delete 4 :
2 3 5 6 7 12 16 18 19 23 26
C:\Users\yashw\Desktop\Summer\Assignment2>_
```

BST-Search

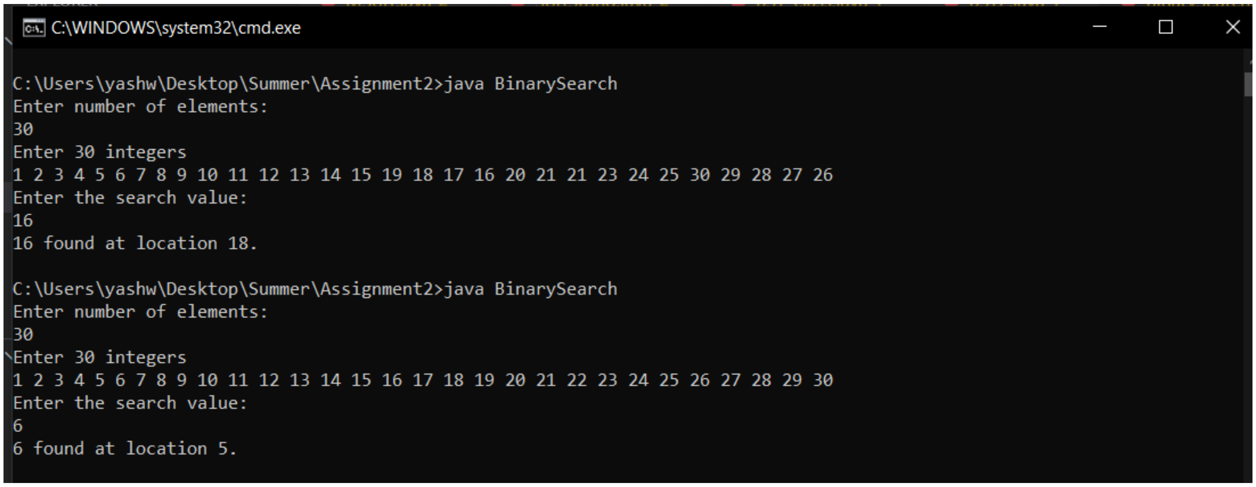
Code

```
import java.util.Scanner;
class BinarySearch
{
  public static void main(String args[])
  {
    int counter, num, item, array[], first, last, middle;
    Scanner input = new Scanner(System.in);
    System.out.println("Enter number of elements:");
    num = input.nextInt();
    array = new int[num];

    System.out.println("Enter " + num + " integers");
    for (counter = 0; counter < num; counter++)
        array[counter] = input.nextInt();

```



```
    System.out.println("Enter the search value:");
    item = input.nextInt();
    first = 0;
    last = num - 1;
    middle = (first + last)/2;

    while( first <= last )
    {
      if ( array[middle] < item )
       first = middle + 1;
      else if ( array[middle] == item )
      {
       System.out.println(item + " found at location " + (middle) + ".");
       break;
      }
      else
      {
         last = middle - 1;
      }
      middle = (first + last)/2;
    }
    if ( first > last )
       System.out.println(item + " is not found.\n");
  }
}
```

Output

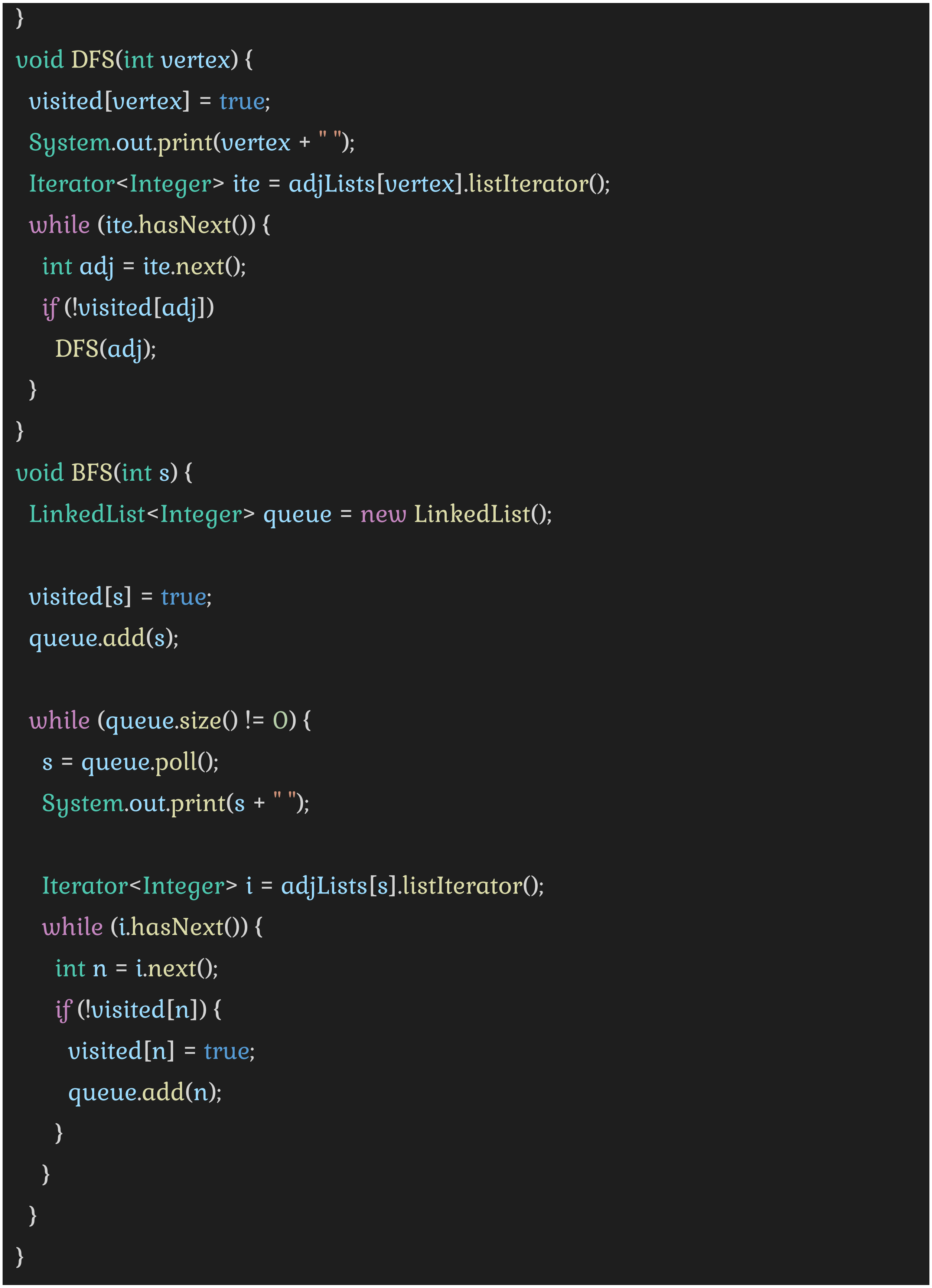


```
C:\Users\yashw\Desktop\Summer\Assignment2>java BinarySearch
Enter number of elements:
30
Enter 30 integers
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 19 18 17 16 20 21 21 23 24 25 30 29 28 27 26
Enter the search value:
16
16 found at location 18.

C:\Users\yashw\Desktop\Summer\Assignment2>java BinarySearch
Enter number of elements:
30
Enter 30 integers
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
Enter the search value:
6
6 found at location 5.
```

Problem-3 :(a) Write a Program to implement the DFS algorithm and print the DFS sequence for the graph below starting with node D.
(b) Write a Program to implement BFS Algorithm and print the BFS sequence starting with node A.

Code

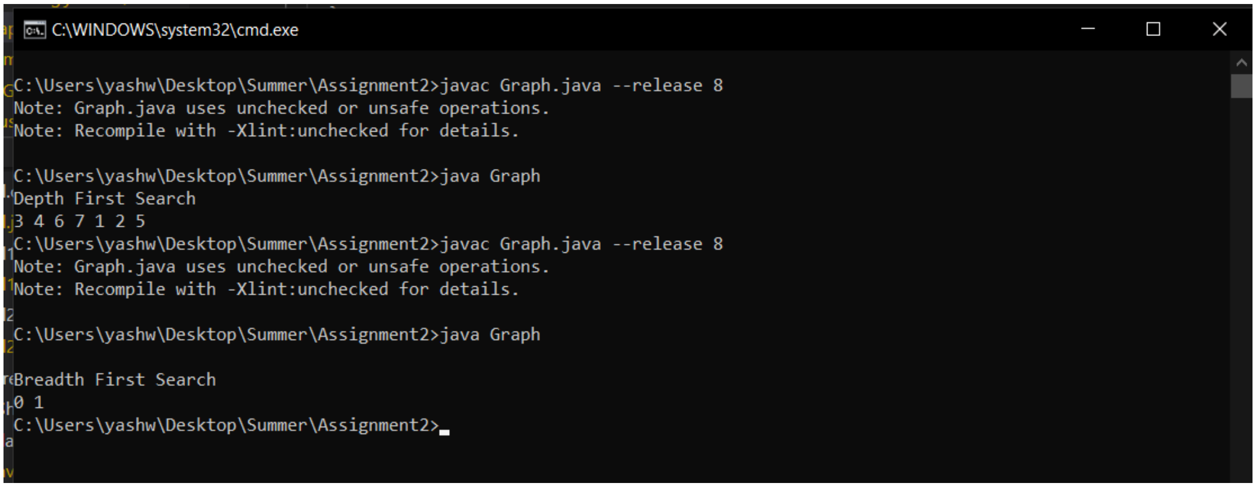
```
import java.util.*;
class Graph {
 private LinkedList<Integer> adjLists[];
 private boolean visited[];

 Graph(int vertices) {
  adjLists = new LinkedList[vertices];
  visited = new boolean[vertices];

  for (int i = 0; i < vertices; i++)
   adjLists[i] = new LinkedList<Integer>();
 }

 void addEdge(int src, int dest) {
  adjLists[src].add(dest);
```



```
}
void DFS(int vertex) {
  visited[vertex] = true;
  System.out.print(vertex + " ");
  Iterator<Integer> ite = adjLists[vertex].listIterator();
  while (ite.hasNext()) {
    int adj = ite.next();
    if (!visited[adj])
      DFS(adj);
  }
}
void BFS(int s) {
  LinkedList<Integer> queue = new LinkedList();

  visited[s] = true;
  queue.add(s);

  while (queue.size() != 0) {
    s = queue.poll();
    System.out.print(s + " ");

    Iterator<Integer> i = adjLists[s].listIterator();
    while (i.hasNext()) {
      int n = i.next();
      if (!visited[n]) {
        visited[n] = true;
        queue.add(n);
      }
    }
  }
}
```

```
 public static void main(String args[]) {
  Graph g = new Graph(10);
// A -0 , B - 1, C-2, D-3, E - 4, F - 5, G -6, H - 7
  g.addEdge(0, 1);
  g.addEdge(7, 1);
  g.addEdge(6, 7);
  g.addEdge(6, 3);
  g.addEdge(4, 6);
  g.addEdge(3, 4);
  g.addEdge(3, 2);
  g.addEdge(3, 5);
  g.addEdge(5, 2);

System.out.println("\nBreadth First Search");
g.BFS(0);
System.out.println("Depth First Search");
g.DFS(3);
 }
}
```

Output



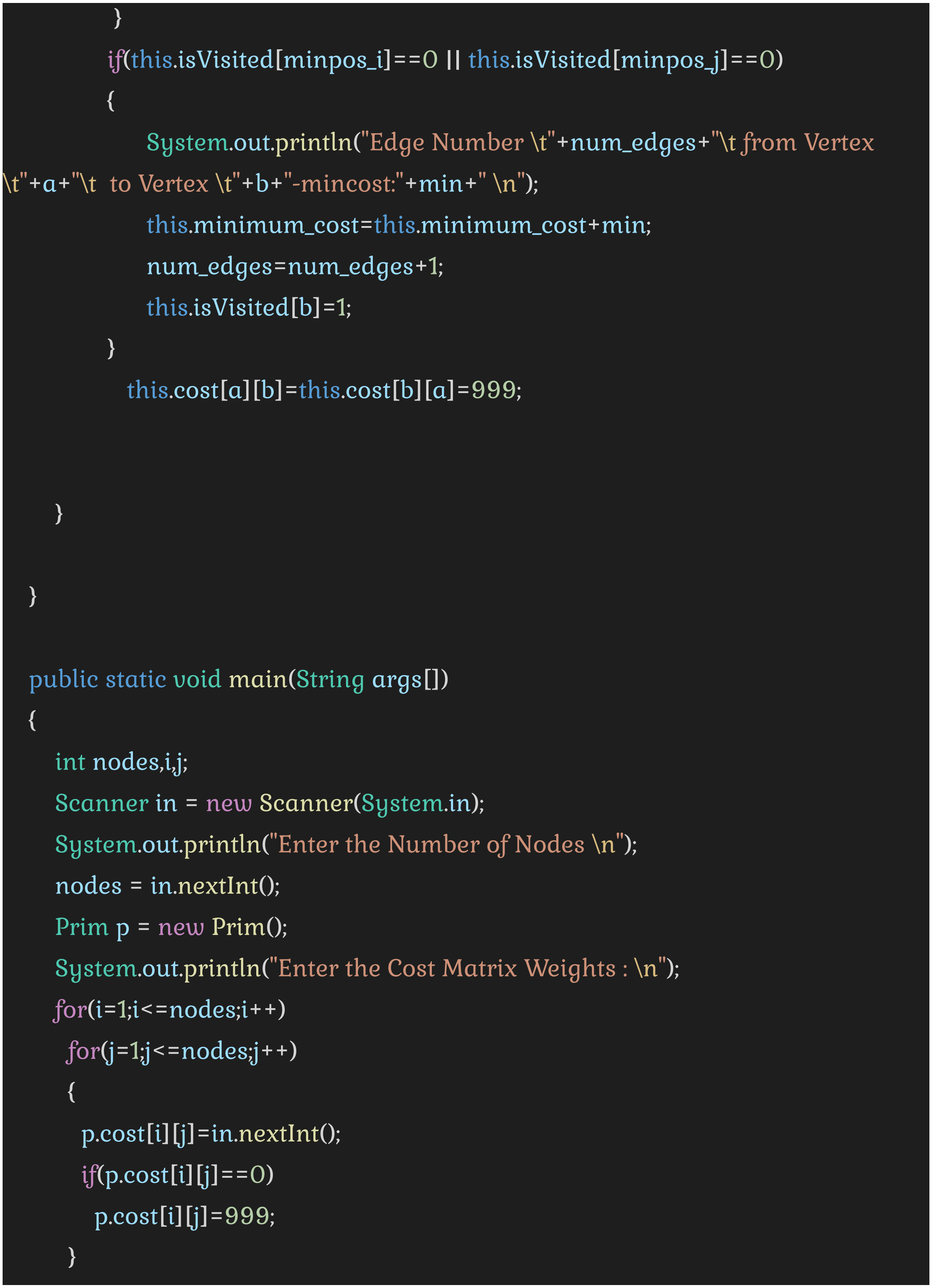
```
C:\Users\yashw\Desktop\Summer\Assignment2>javac Graph.java --release 8
Note: Graph.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

C:\Users\yashw\Desktop\Summer\Assignment2>java Graph
Depth First Search
3 4 6 7 1 2 5
C:\Users\yashw\Desktop\Summer\Assignment2>javac Graph.java --release 8
Note: Graph.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

C:\Users\yashw\Desktop\Summer\Assignment2>java Graph

Breadth First Search
0 1
C:\Users\yashw\Desktop\Summer\Assignment2>
```

Problem-4 :(a) Write a Program to Implement Prim's Algorithm and find the minimum spanning tree for the given graph
(c) Write a Program to implement Kruskal's Algorithm and find the minimum spanning tree for the above graph

Code

Prim's

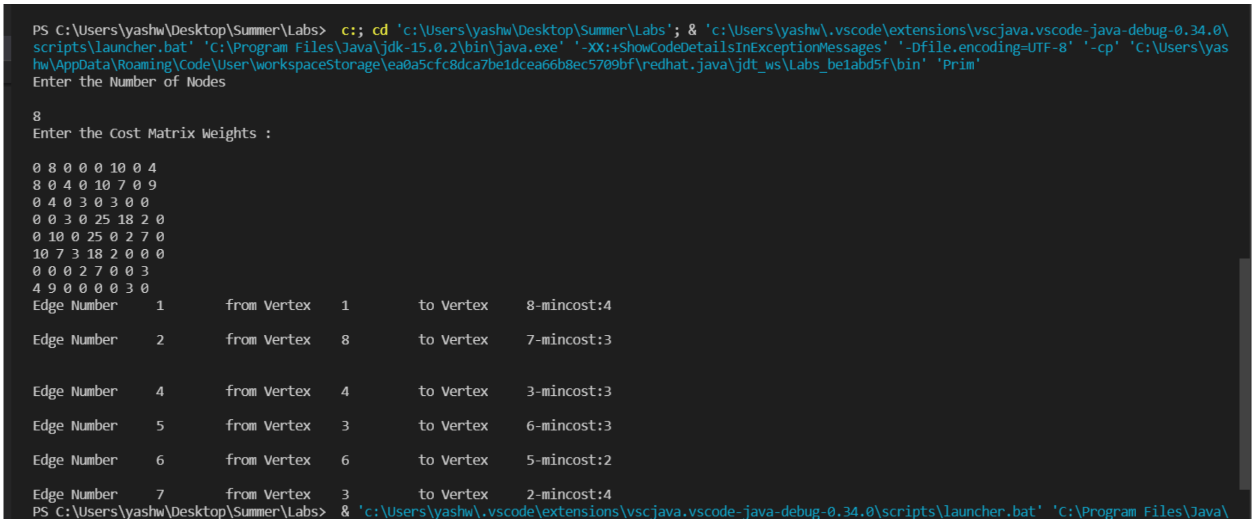
```
import java.util.*;

public class Prim {
    public int isVisited[] = new int[15];
    public int cost[][] = new int[10][10];
    public int minimum_cost;

    public void calc(int n)
    {
        int flag[] = new int[n+1];
        int i,j,min=999,num_edges=1,a=1,b=1,minpos_i=1,minpos_j=1;

        while(num_edges < n)
        {


            for(i=1,min=999;i<=n;i++)
             for(j=1;j<=n;j++)
              if(this.cost[i][j]<min)
                if(this.isVisited[i]!=0)
                {
                     min=this.cost[i][j];
                     a=minpos_i=i;
                     b=minpos_j=j;
```



```
         }
        if(this.isVisited[minpos_i]==0 || this.isVisited[minpos_j]==0)
        {
           System.out.println("Edge Number \t"+num_edges+"\t from Vertex 
\t"+a+"\t  to Vertex \t"+b+"-mincost:"+min+" \n");
           this.minimum_cost=this.minimum_cost+min;
           num_edges=num_edges+1;
           this.isVisited[b]=1;
        }
          this.cost[a][b]=this.cost[b][a]=999;



    }


  }

  public static void main(String args[])
  {
    int nodes,i,j;
    Scanner in = new Scanner(System.in);
    System.out.println("Enter the Number of Nodes \n");
    nodes = in.nextInt();
    Prim p = new Prim();
    System.out.println("Enter the Cost Matrix Weights : \n");
    for(i=1;i<=nodes;i++)
     for(j=1;j<=nodes;j++)
     {
       p.cost[i][j]=in.nextInt();
       if(p.cost[i][j]==0)
         p.cost[i][j]=999;
     }
```

```
        p.isVisited[1]=1;
        p.calc(nodes);
    }
}
```

Output

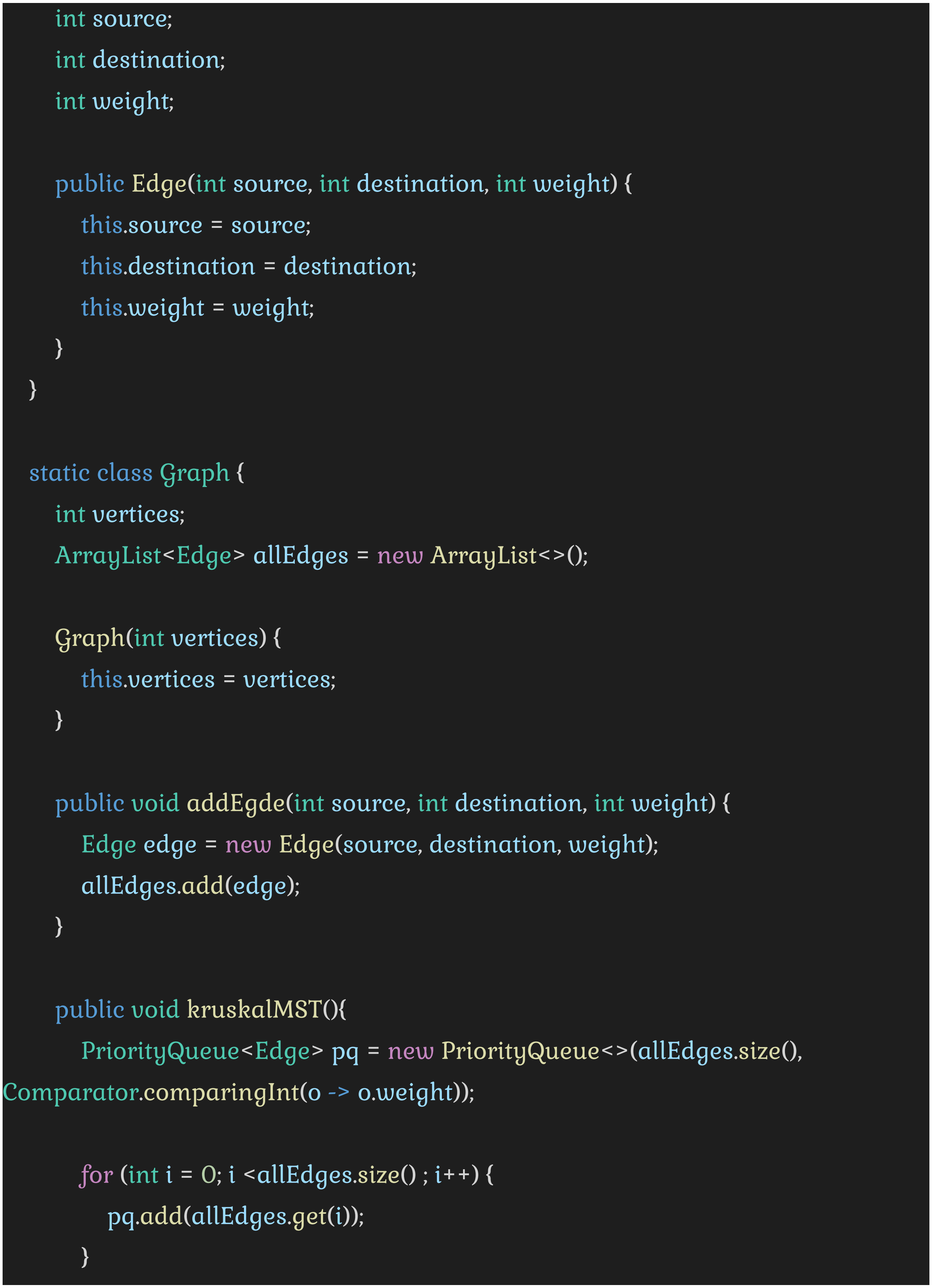


```
PS C:\Users\yashw\Desktop\Summer\Labs>  c:; cd 'c:\Users\yashw\Desktop\Summer\Labs'; & 'c:\Users\yashw\.vscode\extensions\vscjava.vscode-java-debug-0.34.0\
scripts\launcher.bat' 'C:\Program Files\Java\jdk-15.0.2\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-Dfile.encoding=UTF-8' '-cp' 'C:\Users\yas
hw\AppData\Roaming\Code\User\workspaceStorage\ea0a5cfc8dca7be1dcea66b8ec5709bf\redhat.java\jdt_ws\Labs_be1abd5f\bin' 'Prim'
Enter the Number of Nodes

8
Enter the Cost Matrix Weights :

0 8 0 0 0 10 0 4
8 0 4 0 10 7 0 9
0 4 0 3 0 3 0 0
0 0 3 0 25 18 2 0
0 10 0 25 0 2 7 0
10 7 3 18 2 0 0 0
0 0 0 2 7 0 0 3
4 9 0 0 0 0 3 0
Edge Number     1        from Vertex    1         to Vertex     8-mincost:4

Edge Number     2        from Vertex    8         to Vertex     7-mincost:3


Edge Number     4        from Vertex    4         to Vertex     3-mincost:3

Edge Number     5        from Vertex    3         to Vertex     6-mincost:3

Edge Number     6        from Vertex    6         to Vertex     5-mincost:2

Edge Number     7        from Vertex    3         to Vertex     2-mincost:4
PS C:\Users\yashw\Desktop\Summer\Labs>  & 'c:\Users\yashw\.vscode\extensions\vscjava.vscode-java-debug-0.34.0\scripts\launcher.bat' 'C:\Program Files\Java\
```

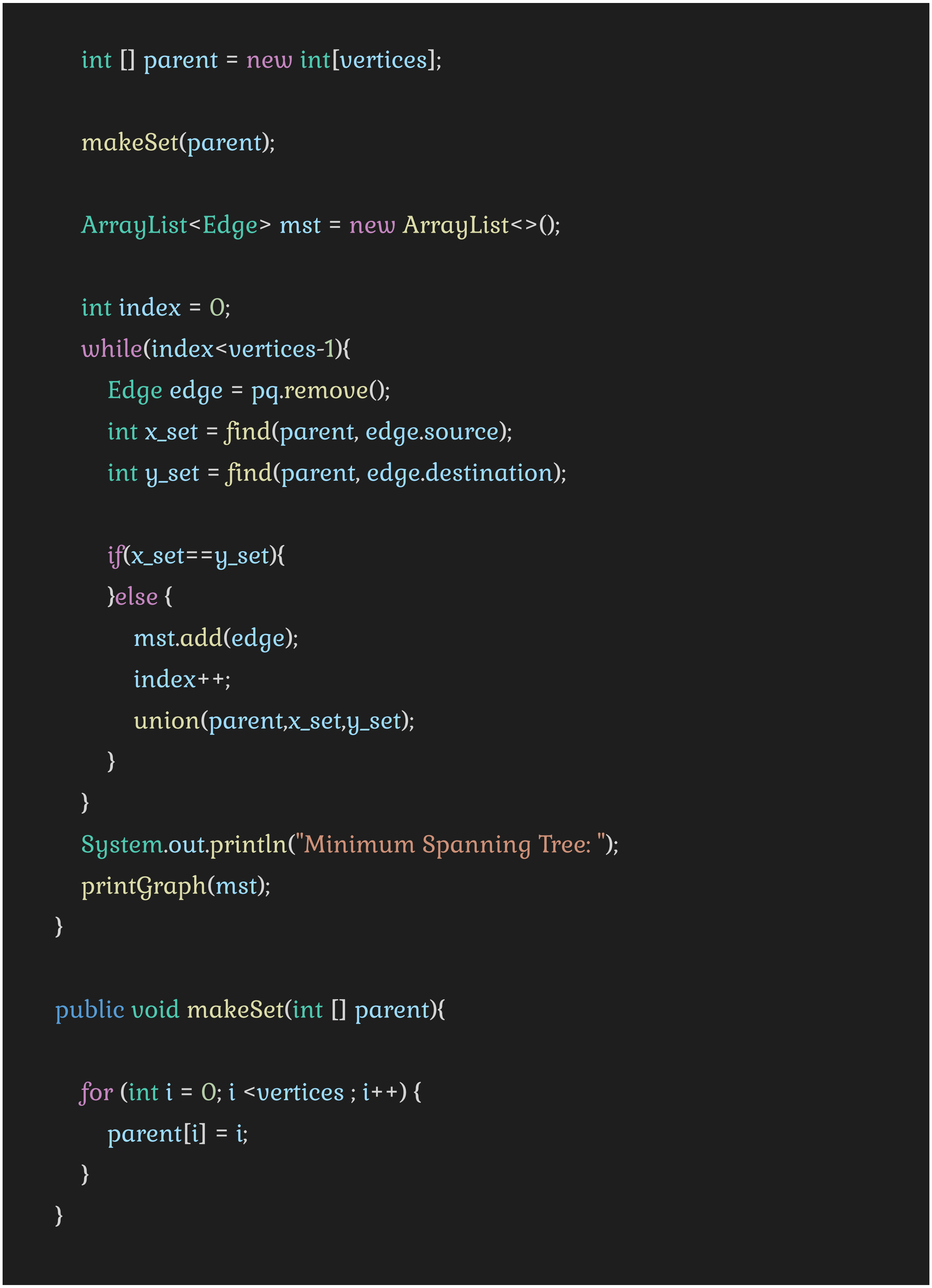
Kruskal’s

Code

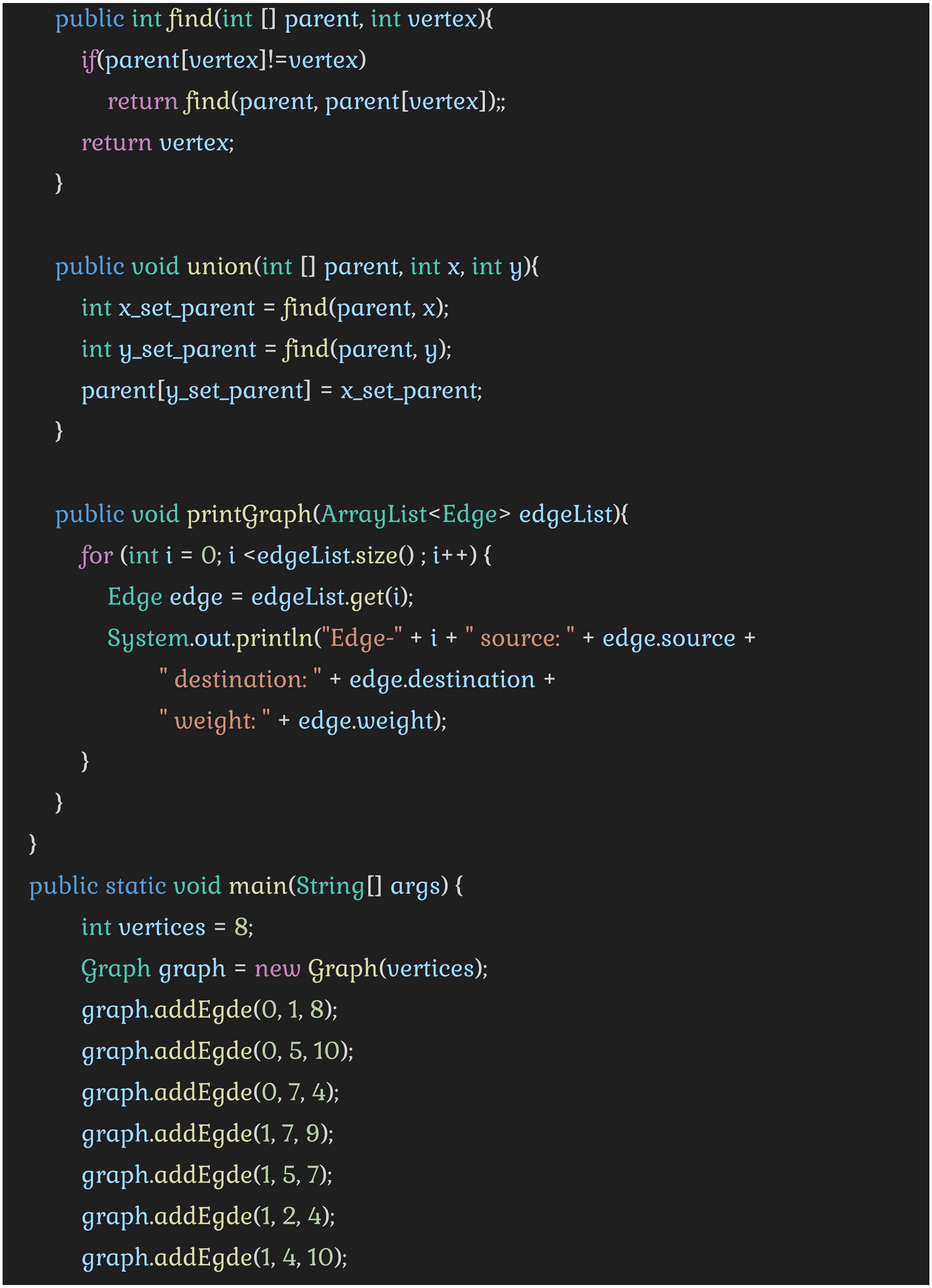
```
import java.util.ArrayList;
import java.util.Comparator;
import java.util.PriorityQueue;

public class KrushkalMST {
    static class Edge {
```



```
        int source;
        int destination;
        int weight;

        public Edge(int source, int destination, int weight) {
            this.source = source;
            this.destination = destination;
            this.weight = weight;
        }
    }

    static class Graph {
        int vertices;
        ArrayList<Edge> allEdges = new ArrayList<>();

        Graph(int vertices) {
            this.vertices = vertices;
        }

        public void addEgde(int source, int destination, int weight) {
            Edge edge = new Edge(source, destination, weight);
            allEdges.add(edge);
        }

        public void kruskalMST(){
            PriorityQueue<Edge> pq = new PriorityQueue<>(allEdges.size(),
Comparator.comparingInt(o -> o.weight));

            for (int i = 0; i <allEdges.size() ; i++) {
                pq.add(allEdges.get(i));
            }
```

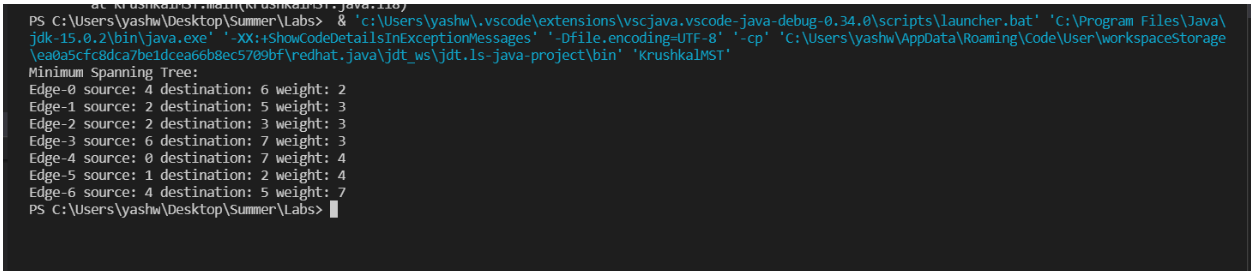


```
    int [] parent = new int[vertices];

    makeSet(parent);

    ArrayList<Edge> mst = new ArrayList<>();

    int index = 0;
    while(index<vertices-1){
        Edge edge = pq.remove();
        int x_set = find(parent, edge.source);
        int y_set = find(parent, edge.destination);

        if(x_set==y_set){
        }else {
            mst.add(edge);
            index++;
            union(parent,x_set,y_set);
        }
    }
    System.out.println("Minimum Spanning Tree: ");
    printGraph(mst);
}

public void makeSet(int [] parent){

    for (int i = 0; i <vertices ; i++) {
        parent[i] = i;
    }
}
```



```
    public int find(int [] parent, int vertex){
        if(parent[vertex]!=vertex)
            return find(parent, parent[vertex]);;
        return vertex;
    }

    public void union(int [] parent, int x, int y){
        int x_set_parent = find(parent, x);
        int y_set_parent = find(parent, y);
        parent[y_set_parent] = x_set_parent;
    }

    public void printGraph(ArrayList<Edge> edgeList){
        for (int i = 0; i <edgeList.size() ; i++) {
            Edge edge = edgeList.get(i);
            System.out.println("Edge-" + i + " source: " + edge.source +
                    " destination: " + edge.destination +
                    " weight: " + edge.weight);
        }
    }
}
public static void main(String[] args) {
        int vertices = 8;
        Graph graph = new Graph(vertices);
        graph.addEgde(0, 1, 8);
        graph.addEgde(0, 5, 10);
        graph.addEgde(0, 7, 4);
        graph.addEgde(1, 7, 9);
        graph.addEgde(1, 5, 7);
        graph.addEgde(1, 2, 4);
        graph.addEgde(1, 4, 10);
        graph.addEgde(2, 3, 3);
        graph.addEgde(2, 5, 3);
        graph.addEgde(3, 5, 18);
        graph.addEgde(3, 4, 10);
        graph.addEgde(3, 6, 25);
        graph.addEgde(4, 6, 2);
        graph.addEgde(4, 5, 7);
        graph.addEgde(6, 7, 3);
        graph.kruskalMST();
    }
}
```

Output



```
        at KrushkalMST.main(KrushkalMST.java:118)
PS C:\Users\yashw\Desktop\Summer\Labs>  & 'c:\Users\yashw\.vscode\extensions\vscjava.vscode-java-debug-0.34.0\scripts\launcher.bat' 'C:\Program Files\Java\
jdk-15.0.2\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-Dfile.encoding=UTF-8' '-cp' 'C:\Users\yashw\AppData\Roaming\Code\User\workspaceStorage
\ea0a5cfc8dca7be1dcea66b8ec5709bf\redhat.java\jdt_ws\jdt.ls-java-project\bin' 'KrushkalMST'
Minimum Spanning Tree:
Edge-0 source: 4 destination: 6 weight: 2
Edge-1 source: 2 destination: 5 weight: 3
Edge-2 source: 2 destination: 3 weight: 3
Edge-3 source: 6 destination: 7 weight: 3
Edge-4 source: 0 destination: 7 weight: 4
Edge-5 source: 1 destination: 2 weight: 4
Edge-6 source: 4 destination: 5 weight: 7
PS C:\Users\yashw\Desktop\Summer\Labs> █
```