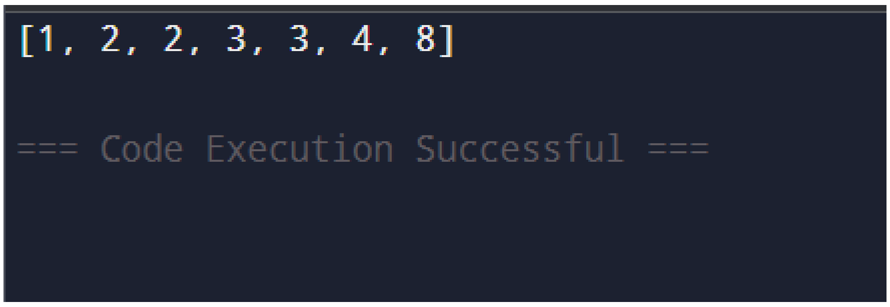
6A.Program :

```
import java.util.*;
public class CountingSort {
   public static void countingSort(int[] arr) {
      int max = Arrays.stream(arr).max().getAsInt();
      int[] count = new int[max + 1];
      for (int num : arr) count[num]++;
      int index = 0;
      for (int i = 0; i <= max; i++) {
         while (count[i]-- > 0) arr[index++] = i;
      }
   }
   public static void main(String[] args) {
      int[] arr = {4, 2, 2, 8, 3, 3, 1};
      countingSort(arr);
      System.out.println(Arrays.toString(arr));
   }
}
```

6A.Output :

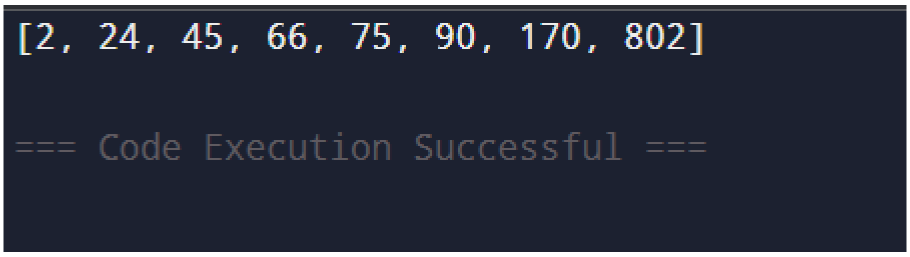


```
[1, 2, 2, 3, 3, 4, 8]

=== Code Execution Successful ===
```

6B.Program :

```
import java.util.*;
public class RadixSort {
    static void radixSort(int[] arr) {
        int max = Arrays.stream(arr).max().getAsInt();
        for (int exp = 1; max / exp > 0; exp *= 10) {
            int[] output = new int[arr.length];
            int[] count = new int[10];
            for (int num : arr) count[(num / exp) % 10]++;
            for (int i = 1; i < 10; i++) count[i] += count[i - 1];
            for (int i = arr.length - 1; i >= 0; i--) {
                int idx = (arr[i] / exp) % 10;
                output[count[idx] - 1] = arr[i];
                count[idx]--;
            }
            System.arraycopy(output, 0, arr, 0, arr.length);
        }
    }
    public static void main(String[] args) {
        int[] arr = {170, 45, 75, 90, 802, 24, 2, 66};
        radixSort(arr);
        System.out.println(Arrays.toString(arr));
    }
}
```

6B.Output :

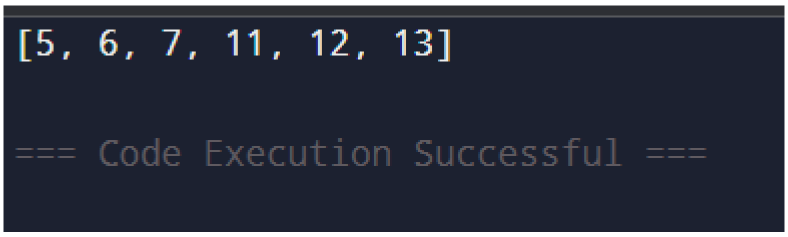


```
[2, 24, 45, 66, 75, 90, 170, 802]


=== Code Execution Successful ===
```

6C.Program :

```
import java.util.*;
public class HeapSort {
    static void heapify(int[] a, int n, int i) {
        int l = 2 * i + 1, r = 2 * i + 2, largest = i;
        if (l < n && a[l] > a[largest]) largest = l;
        if (r < n && a[r] > a[largest]) largest = r;
        if (largest != i) {
            int t = a[i]; a[i] = a[largest]; a[largest] = t;
            heapify(a, n, largest);
        }
    }
    static void heapSort(int[] a) {
        int n = a.length;
        for (int i = n / 2 - 1; i >= 0; i--) heapify(a, n, i);
        for (int i = n - 1; i > 0; i--) {
            int t = a[0]; a[0] = a[i]; a[i] = t;
            heapify(a, i, 0);
        }
    }
    public static void main(String[] args) {
        int[] a = {12, 11, 13, 5, 6, 7};
        heapSort(a);
        System.out.println(Arrays.toString(a));
    }
}
```

6C.Output :

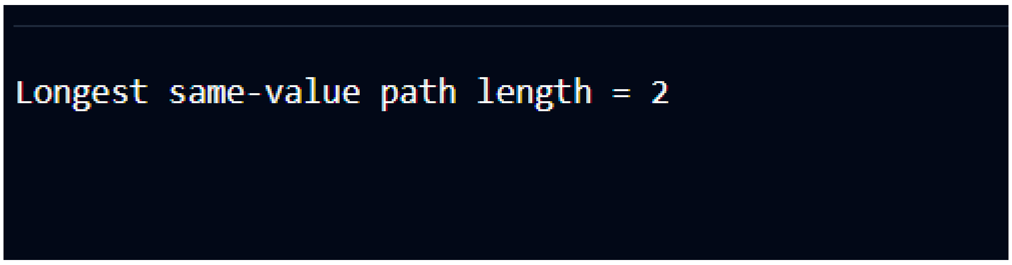


```
[5, 6, 7, 11, 12, 13]

=== Code Execution Successful ===
```

7A.Program :

```
import java.util.*;
class Node { int val; Node left, right; Node(int v){val=v;} }
public class LongestUnivaluePath {
   static int ans=0;
   static int dfs(Node r){
      if(r==null) return 0;
      int l=dfs(r.left), rt=dfs(r.right);
      int lp=r.left!=null&&r.left.val==r.val?l+1:0;
      int rp=r.right!=null&&r.right.val==r.val?rt+1:0;
      ans=Math.max(ans,lp+rp);
      return Math.max(lp,rp);
   }
   public static void main(String[] a){
      Node r=new Node(5);
      r.left=new Node(4); r.right=new Node(5);
      r.left.left=new Node(1); r.left.right=new Node(1);
      r.right.right=new Node(5);
      dfs(r);
      System.out.println(ans);
   }
}
```

7A.Output :

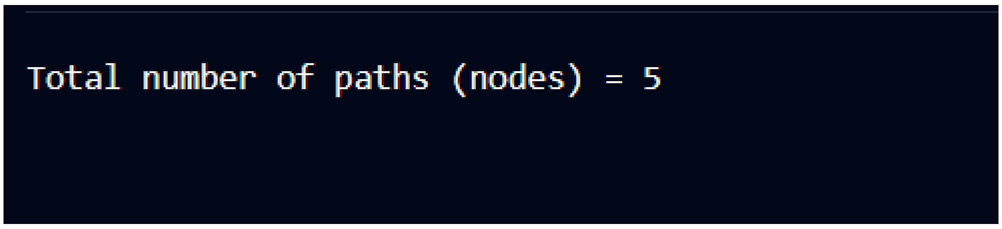


```
Longest same-value path length = 2
```

7B.Program :

```
class Node2 {
   int val; Node2 left, right;
   Node2(int v){ val=v; }
}
public class CountPaths {
   static int count(Node2 r){
      if(r==null) return 0;
      return 1 + count(r.left) + count(r.right);
   }
   public static void main(String[] a){
      Node2 r=new Node2(1);
      r.left=new Node2(2); r.right=new Node2(3);
      r.left.left=new Node2(4); r.left.right=new Node2(5);
      int total = count(r);
      System.out.println("Total number of paths (nodes) = " + total);
   }
}
```

7B.Output :

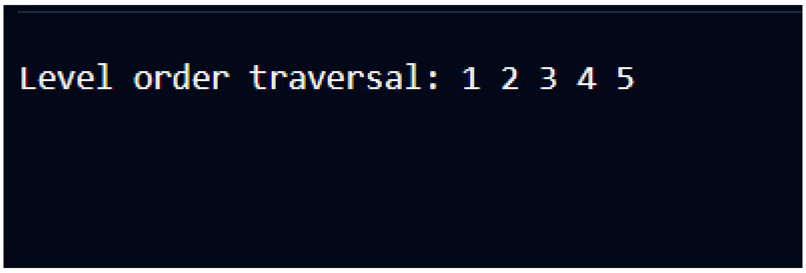


```
Total number of paths (nodes) = 5
```

7C.Program :

```
import java.util.*;
class Node3 {
   int val; Node3 left, right;
   Node3(int v){ val=v; }
}
public class LevelOrder {
   static void levelOrder(Node3 r){
      if(r==null) return;
      Queue<Node3> q=new LinkedList<>();
      q.add(r);
      System.out.print("Level order traversal: ");
      while(!q.isEmpty()){
         Node3 n=q.poll();
         System.out.print(n.val+" ");
         if(n.left!=null) q.add(n.left);
         if(n.right!=null) q.add(n.right);
      }
      System.out.println();
   }
   public static void main(String[] a){
      Node3 r=new Node3(1);
      r.left=new Node3(2); r.right=new Node3(3);
      r.left.left=new Node3(4); r.left.right=new Node3(5);
      levelOrder(r);
   }
}
```

7C.Output :

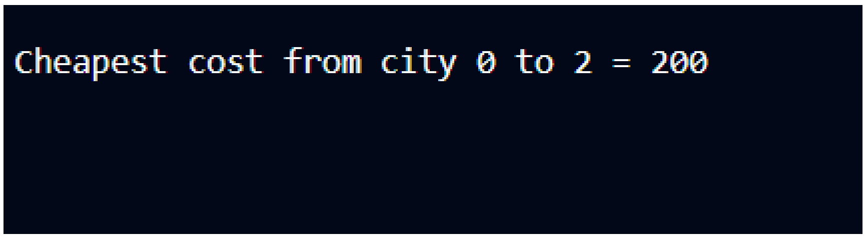


```
Level order traversal: 1 2 3 4 5
```

8A.Program :

```
import java.util.*;
public class CheapestFlight {
   static int findCheapest(int n, int[][] flights, int src, int dst){
      Map<Integer,List<int[]>> g=new HashMap<>();
      for(int[] f:flights) g.computeIfAbsent(f[0],k->new ArrayList<>()).add(new
int[]{f[1],f[2]});
      int[] dist=new int[n]; Arrays.fill(dist,Integer.MAX_VALUE); dist[src]=0;
      PriorityQueue<int[]> pq=new PriorityQueue<>(Comparator.comparingInt(a->a[1]));
      pq.add(new int[]{src,0});
      while(!pq.isEmpty()){
         int[] cur=pq.poll();
         if(cur[0]==dst) return cur[1];
         if(cur[1]>dist[cur[0]]) continue;
         for(int[] e:g.getOrDefault(cur[0],new ArrayList<>())){
            int next=e[0], nd=cur[1]+e[1];
            if(nd<dist[next]){ dist[next]=nd; pq.add(new int[]{next,nd}); }
         }
      }
      return -1;
   }
   public static void main(String[] a){
      int[][] flights={{0,1,100},{1,2,100},{0,2,500}};
      int cost=findCheapest(3,flights,0,2);
      System.out.println("Cheapest cost from city 0 to 2 = "+cost);
   }
}
```

8A.Output :

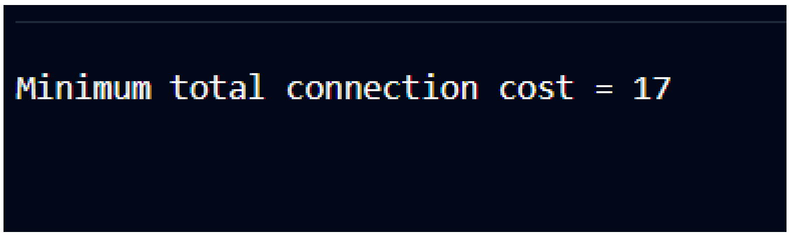


```
Cheapest cost from city 0 to 2 = 200
```

8B.Program :

```
public class ConnectGroups {
    static int connectTwoGroups(int[][] cost){
        int m=cost.length,n=cost[0].length;
        int[] minB=new int[n];
        for(int j=0;j<n;j++){ int min=Integer.MAX_VALUE; for(int i=0;i<m;i++)
min=Math.min(min,cost[i][j]); minB[j]=min; }
        int[] dp=new int[1<<n]; for(int i=1;i<(1<<n);i++) dp[i]=100000000;
        for(int i=0;i<m;i++){
            int[] ndp=new int[1<<n]; Arrays.fill(ndp,100000000);
            for(int mask=0;mask<(1<<n);mask++)
                for(int j=0;j<n;j++){
                    int nmask=mask|(1<<j);
                    ndp[nmask]=Math.min(ndp[nmask],dp[mask]+cost[i][j]);
                }
            dp=ndp;
        }
        int res=Integer.MAX_VALUE;
        for(int mask=0;mask<(1<<n);mask++){
            int s=dp[mask];
            for(int j=0;j<n;j++) if((mask&(1<<j))==0) s+=minB[j];
            res=Math.min(res,s);
        }
        return res;
    }
    public static void main(String[] a){
        int[][] cost={{15,96},{36,2}};
        System.out.println("Minimum total connection cost = "+connectTwoGroups(cost));
    }
}
```

8B.Output :

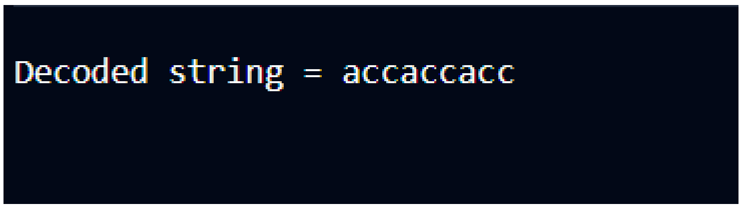


```
Minimum total connection cost = 17
```

8C.Program :

```
import java.util.*;
public class DecodeString {
   static String decode(String s){
      Stack<Integer> count=new Stack<>();
      Stack<StringBuilder> res=new Stack<>();
      StringBuilder cur=new StringBuilder(); int k=0;
      for(char c:s.toCharArray()){
         if(Character.isDigit(c)) k=k*10+(c-'0');
         else if(c=='['){ count.push(k); res.push(cur); cur=new StringBuilder(); k=0; }
         else if(c==']'){
            StringBuilder tmp=res.pop();
            int times=count.pop();
            while(times-->0) tmp.append(cur);
            cur=tmp;
         } else cur.append(c);
      }
      return cur.toString();
   }
   public static void main(String[] a){
      String s="3[a2[c]]";
      System.out.println("Decoded string = "+decode(s));
   }
}
```

8C.Output :

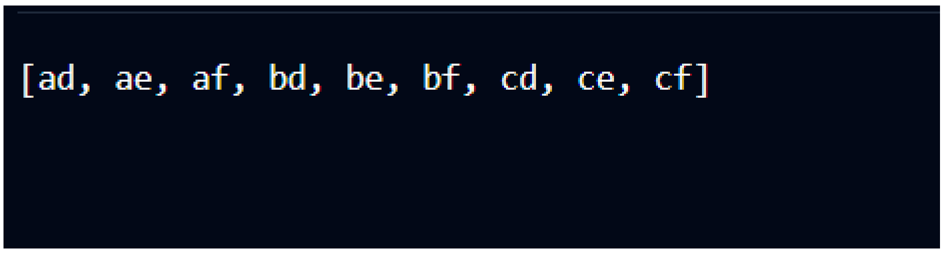


```
Decoded string = accaccacc
```

9A.Program :

```
import java.util.*;
public class LetterCombinations {
   static Map<Character,String> m=Map.of(
      '2',"abc",'3',"def",'4',"ghi",'5',"jkl",
      '6',"mno",'7',"pqrs",'8',"tuv",'9',"wxyz"
   );
   static void backtrack(List<String> r,StringBuilder c,String s,int i){
      if(i==s.length()){ r.add(c.toString()); return; }
      for(char ch:m.get(s.charAt(i)).toCharArray()){
         c.append(ch); backtrack(r,c,s,i+1); c.deleteCharAt(c.length()-1);
      }
   }
   public static void main(String[] a){
      String s="23"; List<String> r=new ArrayList<>();
      backtrack(r,new StringBuilder(),s,0);
      System.out.println(r);
   }
}
```

9A.Output :

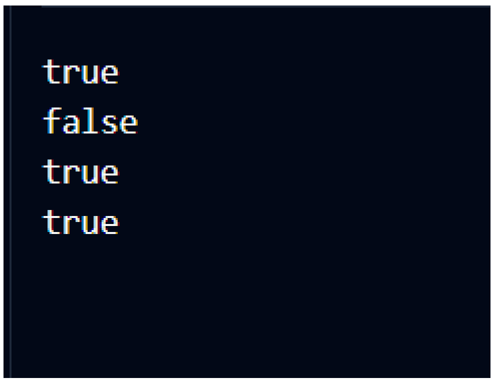


```
[ad, ae, af, bd, be, bf, cd, ce, cf]
```

9B.Program :

```
public class RegexMatching {
   static boolean match(String s,String p,int i,int j){
      if(j==p.length()) return i==s.length();
      boolean f=i<s.length()&&(s.charAt(i)==p.charAt(j)||p.charAt(j)=='.');
      if(j+1<p.length()&&p.charAt(j+1)=='*')
         return match(s,p,i,j+2)||(f&&match(s,p,i+1,j));
      else return f&&match(s,p,i+1,j+1);
   }
   public static void main(String[] a){
      System.out.println(match("aa","a*",0,0));
      System.out.println(match("mississippi","mis*is*p*.",0,0));
      System.out.println(match("ab",".*",0,0));
      System.out.println(match("aab","c*a*b",0,0));
   }
}
```

9B.Output :

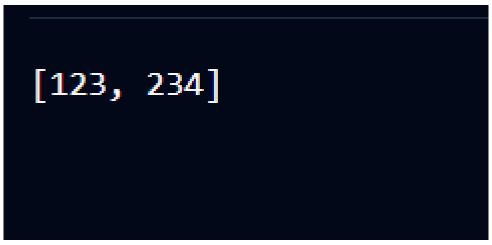


```
true
false
true
true
```

9C.Program :

```
import java.util.*;
public class SequentialDigits {
   static void gen(int d,int n,int l,int h,List<Integer> r){
      if(n>=l&&n<=h) r.add(n);
      if(n>h||d>9) return;
      gen(d+1,n*10+d,l,h,r);
   }
   public static void main(String[] a){
      List<Integer> r=new ArrayList<>();
      for(int i=1;i<=9;i++) gen(i,0,100,300,r);
      Collections.sort(r);
      System.out.println(r);
   }
}
```

9C.Output :

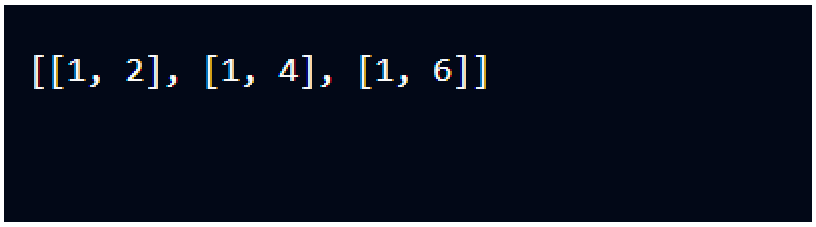


```
[123, 234]
```

10A.Program :

```
import java.util.*;

public class KSmallestPairs {

   public static void main(String[] a){

      int[] n1={1,7,11}, n2={2,4,6}; int k=3;

      PriorityQueue<int[]> pq=new PriorityQueue<>(Comparator.comparingInt(x-
>n1[x[0]]+n2[x[1]]));

      for(int i=0;i<Math.min(k,n1.length);i++) pq.add(new int[]{i,0});

      List<List<Integer>> r=new ArrayList<>();

      while(k-->0&&!pq.isEmpty()){

         int[] p=pq.poll();

         r.add(Arrays.asList(n1[p[0]],n2[p[1]]));

         if(p[1]+1<n2.length) pq.add(new int[]{p[0],p[1]+1});

      }

      System.out.println(r);

   }

}
```

10A.Output :

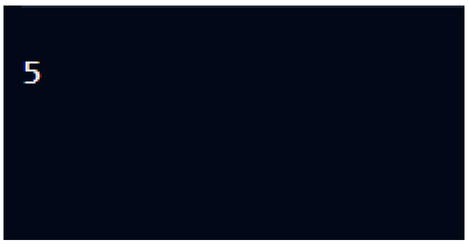


```
[[1, 2], [1, 4], [1, 6]]
```

10B.Program :

```
import java.util.*;
public class KthLargest {
   public static void main(String[] a){
      int[] n={3,2,1,5,6,4}; int k=2;
      PriorityQueue<Integer> pq=new PriorityQueue<>();
      for(int x:n){ pq.add(x); if(pq.size()>k) pq.poll(); }
      System.out.println(pq.peek());
   }
}
```

10B.Output :

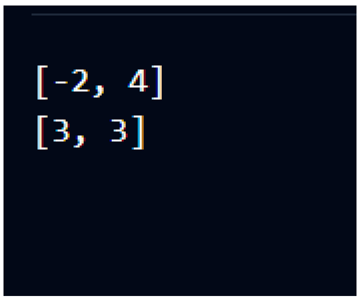


```
5
```

10C.Program :

```
import java.util.*;

public class KClosestPoints {

   public static void main(String[] a){

      int[][] p={{3,3},{5,-1},{-2,4}}; int k=2;

      PriorityQueue<int[]> pq=new PriorityQueue<>((x,y)->(y[0]*y[0]+y[1]*y[1])-
(x[0]*x[0]+x[1]*x[1]));

      for(int[] pt:p){ pq.add(pt); if(pq.size()>k) pq.poll(); }

      int[][] r=new int[k][2];

      for(int i=0;i<k;i++) r[i]=pq.poll();

      for(int[] x:r) System.out.println(Arrays.toString(x));

   }

}
```

10C.Output :



```
[-2, 4]
[3, 3]
```