DEPARTMENT OF COMPUTER ENGINEERING AND APPLICATIONS

PRACTICAL FILE

ON

DESIGN AND ANALYSIS OF ALGORITHMS (BCSC 0807)

NAME : Jagrati Garg

ROLL NO. : 201500309

SECTION : A

CLASS ROLL NO. : 28

Submitted to : Dr.Pawan Kumar Mishra

Submitted by : Jagrati Garg

Signature :

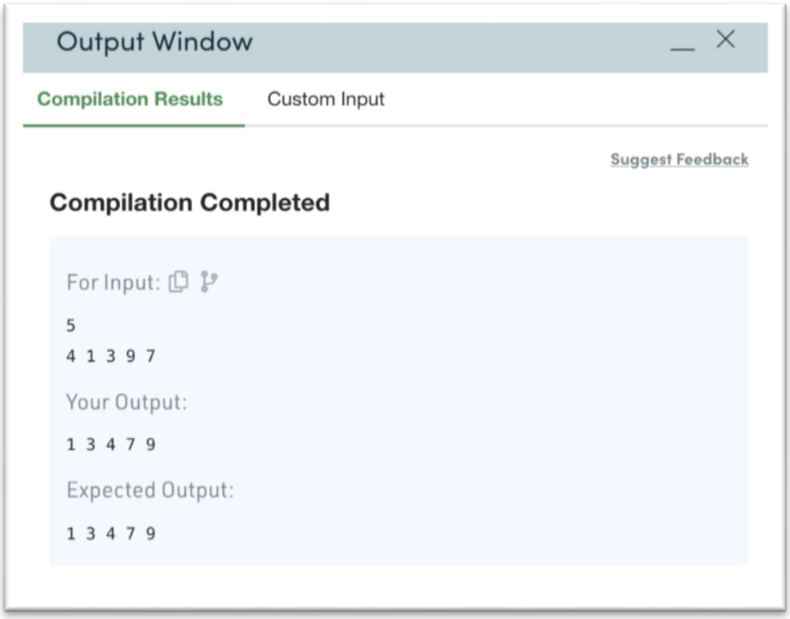
Name :Jagrati Garg (28)

INDEX

Name :Jagrati Garg (28)

INSERTION SORT

```
class Solution
{
    static void insert(int arr[],int i)
    {
              // Your code hereint
              key=arr[i]; int j=i-1;
              while(j>=0 && arr[j]>key)
              {
                    arr[j+1]=arr[j];j--;
              }
              arr[j+1]=key;
    }
    //Function to sort the array using insertion sort algorithm.public void insertionSort(int arr[],
    int n)
    {
            //code here
            for(int i=0; i<n;i++)
            {
                    insert(arr,i);
            }
    }
```



Compilation Completed

For Input:

```
5
4 1 3 9 7
```

Your Output:

```
1 3 4 7 9
```

Expected Output:

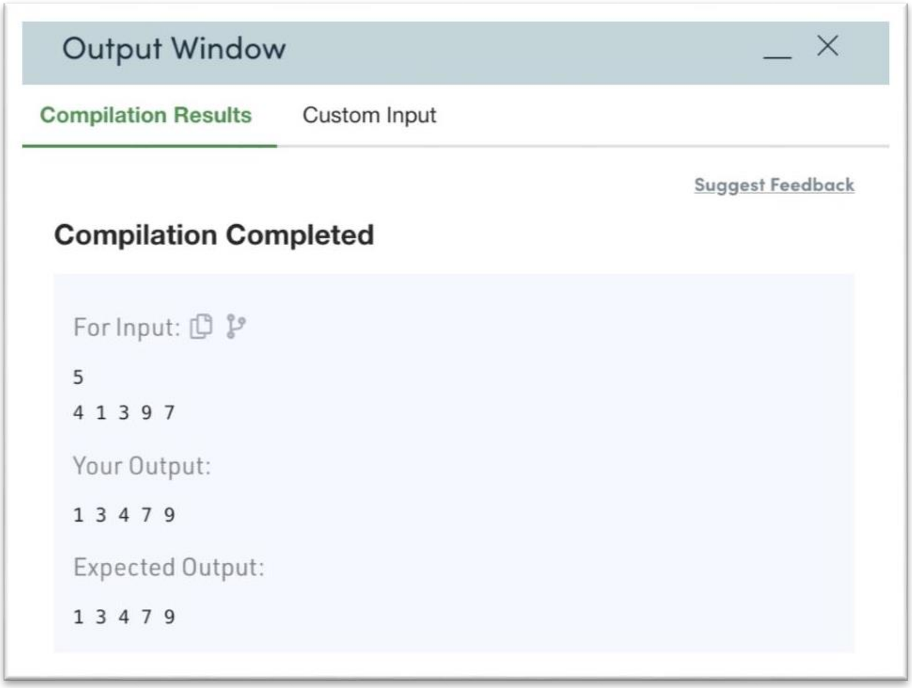
```
1 3 4 7 9
```

```
}
```

Name :Jagrati Garg (28)

BUBBLE SORT

```
class Solution
{
            //Function to sort the array using bubble sort algorithm.public static void
        bubbleSort(int arr[], int n)
            {
            //code here
            for(int i=0; i<arr.length-1; i++)
            {
                        for(int j=0; j<arr.length-1-i; j++)
                        {
                        if(arr[j]>arr[j+1]){
                                    int temp = arr[j]; arr[j] =
                                    arr[j+1];arr[j+1] = temp;
                        }
                        }
            }
            }
}
```



Compilation Completed

For Input:

```
5
4 1 3 9 7
```

Your Output:

```
1 3 4 7 9
```

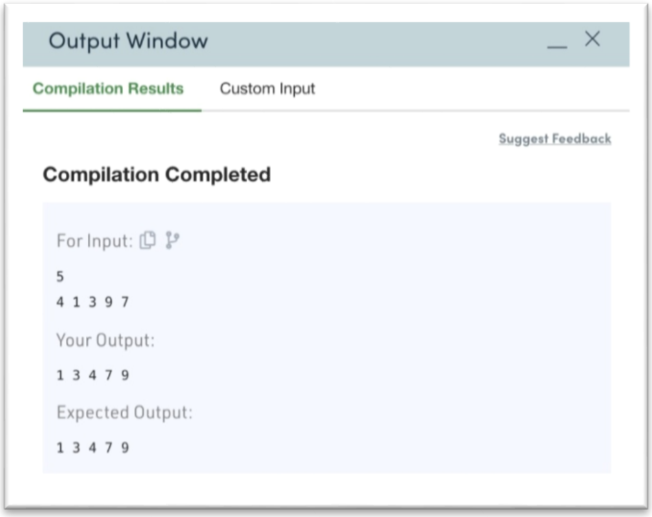
Expected Output:

```
1 3 4 7 9
```

Name :Jagrati Garg (28)

SELECTION SORT

```
class Solution
{
        int  select(int arr[], int i)
        {
           // code here such that selectionSort() sorts arr[]int mini=i;
           for(int j=i; j<=arr.length-1; j++)
           {
                 if(arr[j]<arr[mini])
                 {
                       mini=j;
                 }
           }
           int temp=arr[mini];
           arr[mini]=arr[i];
           arr[i]=temp; return mini;

        }
        void selectionSort(int arr[], int n)
        {
              //code here
              for(int i=0; i<=n-2;i++)
              {
                    select(arr,i);
              }
        }
```



Compilation Completed

```
For Input:
5
4 1 3 9 7
Your Output:
1 3 4 7 9
Expected Output:
1 3 4 7 9
```

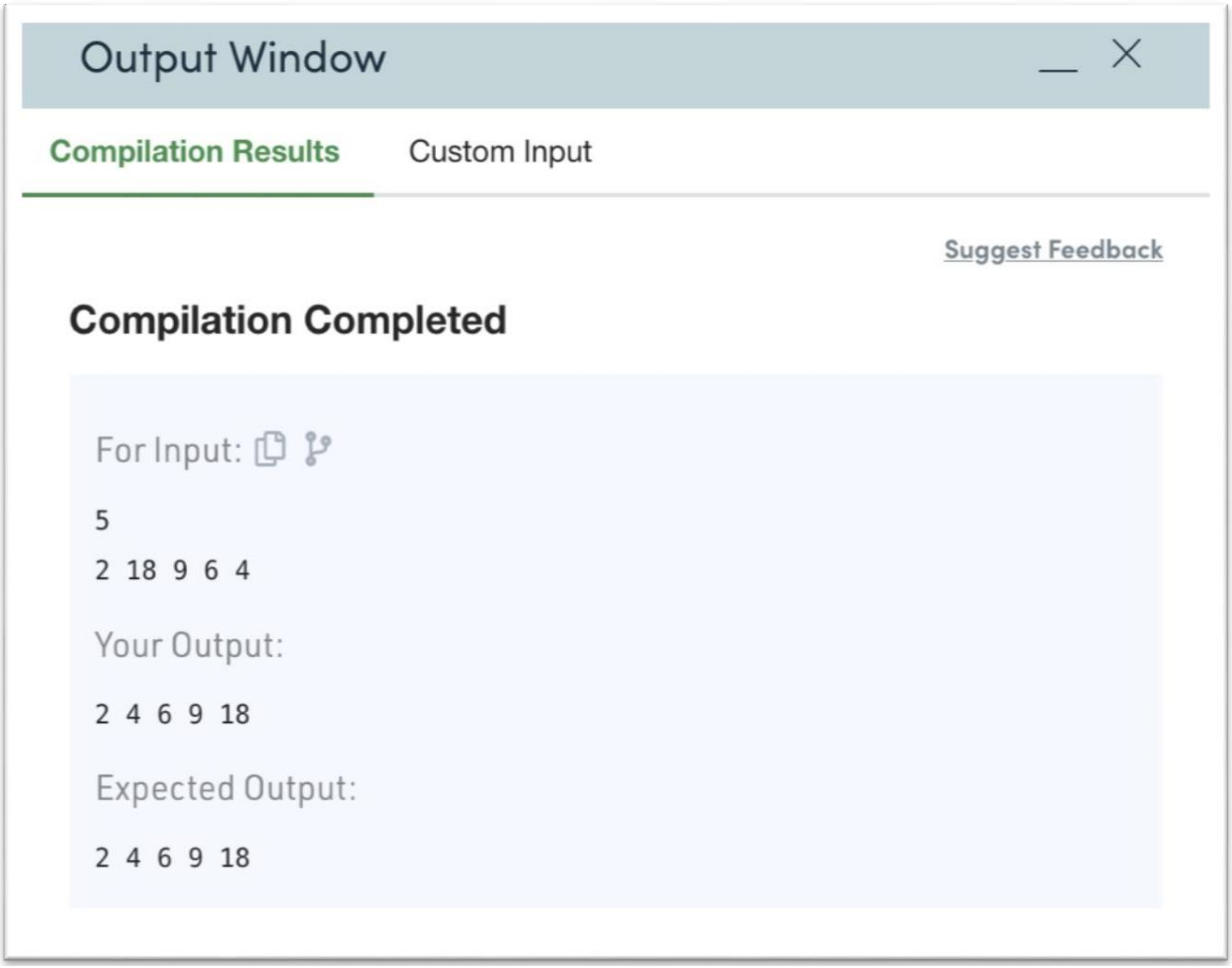
```
}
```

Name :Jagrati Garg (28)

QUICK SORT

```
class Solution
{
    static void swap(int[] arr, int i, int j)
    {
    int temp = arr[i];arr[i] =
    arr[j]; arr[j] = temp;
    }
    //Function to sort an array using quick sort algorithm.static void quickSort(int arr[], int
    low, int high)
    {
    // code here if (low <
    high)
    {
        int pi = partition(arr, low, high);quickSort(arr, low, pi -
        1); quickSort(arr, pi + 1, high);
    }
    }
    static int partition(int arr[], int low, int high)
    {
    // your code here
    int pivot = arr[high];int i = (low -
    1);

    for (int j = low; j <= high - 1; j++) {if (arr[j] < pivot) {
        i++;
        swap(arr, i, j);
        }
    }
    swap(arr, i + 1, high);return (i +
    1);
    }
}
```

Name :Jagrati Garg (28)



Compilation Completed

For Input:

```
5
2 18 9 6 4
```

Your Output:

```
2 4 6 9 18
```

Expected Output:

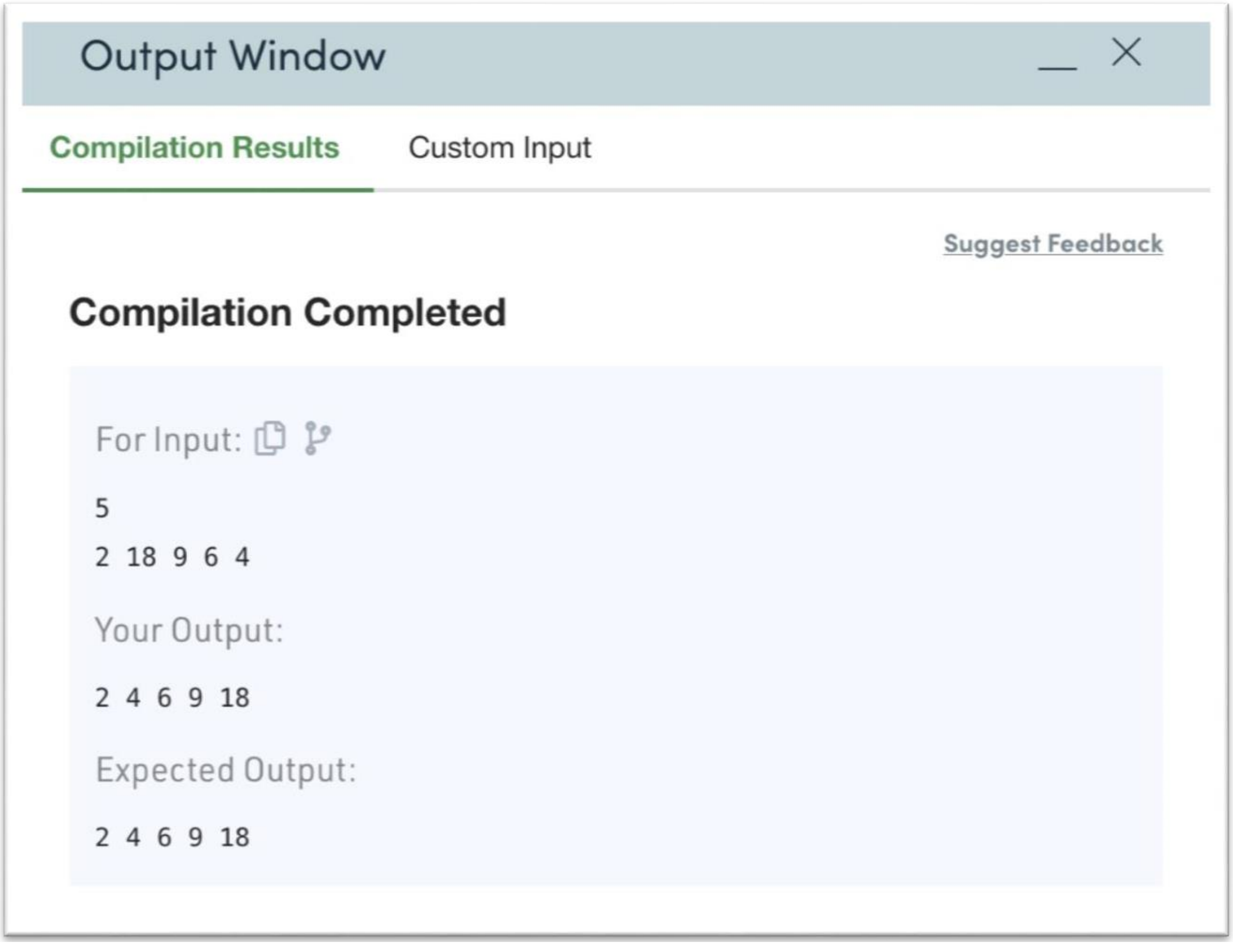
```
2 4 6 9 18
```

Name :Jagrati Garg (28)

MERGE SORT

```
class Solution
{
        void merge(int arr[], int l, int mid, int r)
        {
        // Your code here
        int merged[]=new int[r-l+1];int i1=l;
        int i2=mid+1;int
        x=0;
        while(i1<=mid && i2<=r){ if(arr[i1]<=arr[i2]){
                        merged[x++]=arr[i1++];
                }
                else
                        merged[x++]=arr[i2++];
        }
        while(i1<=mid){
                merged[x++]=arr[i1++];

        }
        while(i2<=r){
                merged[x++]=arr[i2++];

        }
        for(inti=0,j=l;i<merged.length;i++,j++){arr[j]=merged[i];
        }
        }
        void mergeSort(int arr[], int l, int r)
        {
        if(l>=r){
                return;
        }
        int mid=(l+r)/2;
        mergeSort(arr,l,mid);
        mergeSort(arr,mid+1,r);
        merge(arr,l,mid,r);

}
}
```

Name :Jagrati Garg (28)



Compilation Completed

For Input:

```
5
2 18 9 6 4
```

Your Output:

```
2 4 6 9 18
```

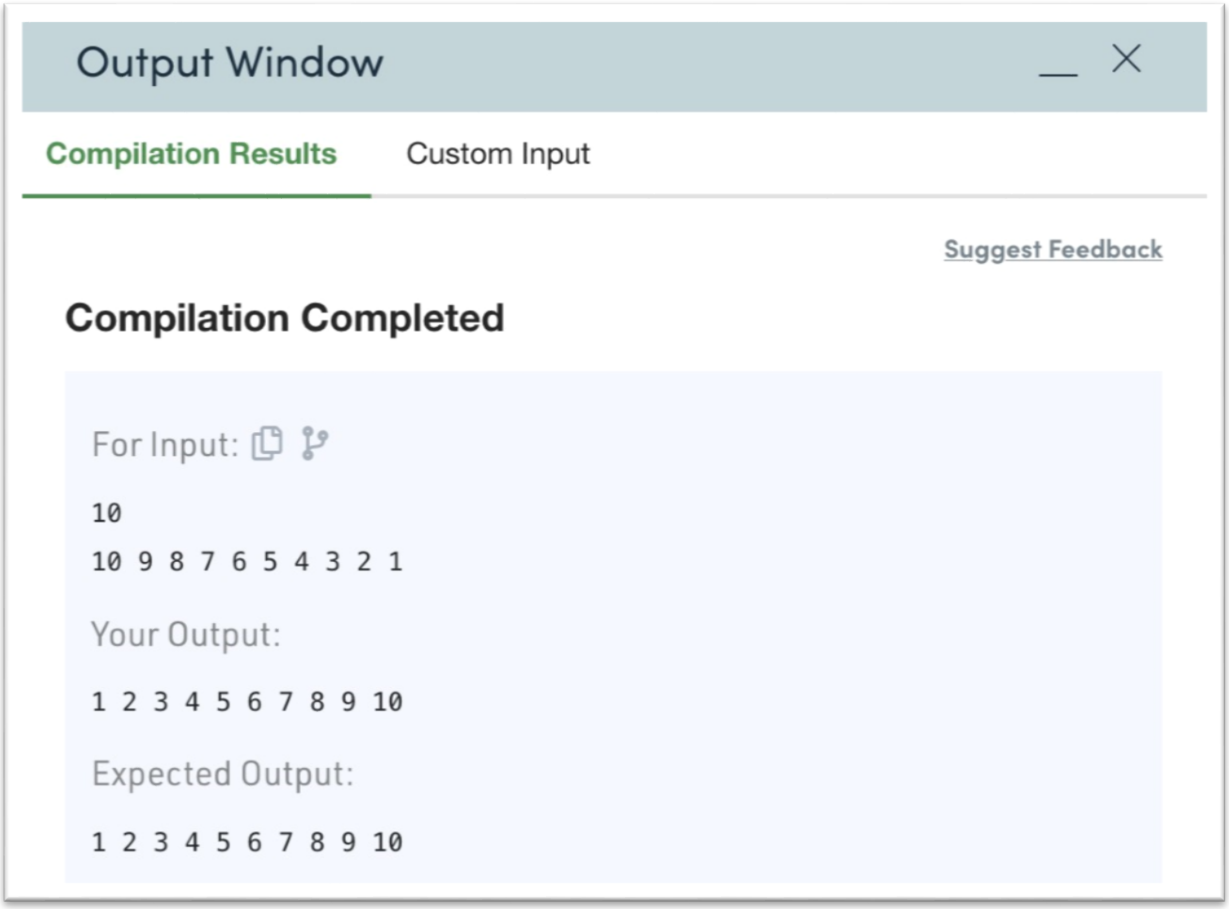
Expected Output:

```
2 4 6 9 18
```

HEAP SORT

```
class Solution
{
    //Function to build a Heap from array.void buildHeap(int
    arr[], int n)
    {
        for(int i=n/2-1;i>=0;--i){heapify(arr, n,
            i);
        }
    }
    //Heapify function to maintain heap property.void heapify(int arr[],
    int n, int i)
    {
        int l = 2*i+1; int r =
        2*i+2; int largest= i;
        if(l<n && arr[l]> arr[largest]){largest    = l;
        }
        if(r<n && arr[r]> arr[largest]){largest    = r;
        }
        if(i != largest){
            int temp = arr[i]; arr[i] =
            arr[largest];arr[largest] = temp;

            heapify(arr, n, largest);
        }
    }

    //Function to sort an array using Heap Sort.public void heapSort(int
    arr[], int n)
    {
        buildHeap(arr,n); for(int i=n-
        1;i>0;--i){
            int temp = arr[0];arr[0]=
            arr[i]; arr[i] = temp;
```

```
                    heapify(arr, i, 0);
                }
            }
        }
```



Compilation Completed

For Input:

```
10
10 9 8 7 6 5 4 3 2 1
```

Your Output:

```
1 2 3 4 5 6 7 8 9 10
```

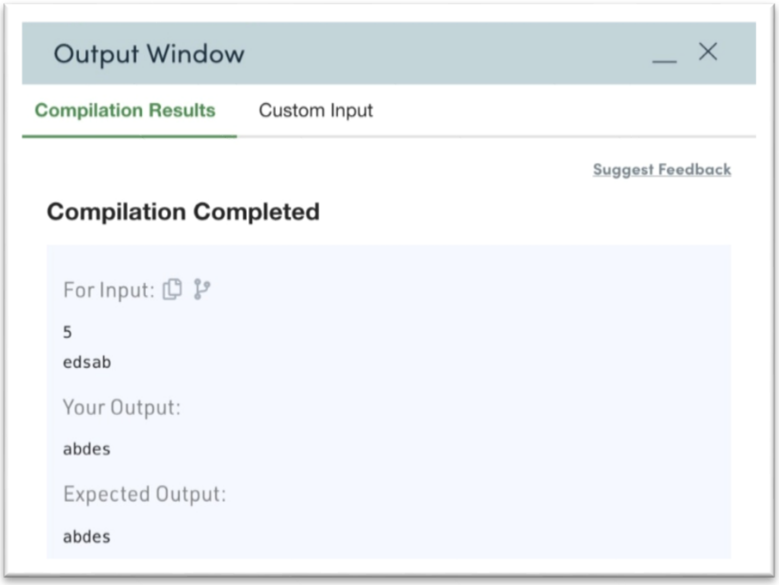
Expected Output:

```
1 2 3 4 5 6 7 8 9 10
```

Name :Jagrati Garg (28)

COUNTING SORT

```
class Solution
{
public static String countSort(String arr)
        {
        int freq[]=new int[26]; for(int
        i=0;i<arr.length();i++)freq[arr.charAt(i)-
        'a']++; String w="";
        for(int i=0;i<26;i++)
        {
                if(freq[i]!=0)
                {       while(freq[i]!=0)
                {
                w+=(char)(i+'a');
                freq[i]--;
                }

                }
        }
        return w;
        // code here
        }
}
```



Compilation Completed

For Input:

```
5
edsab
```

Your Output:

```
abdes
```

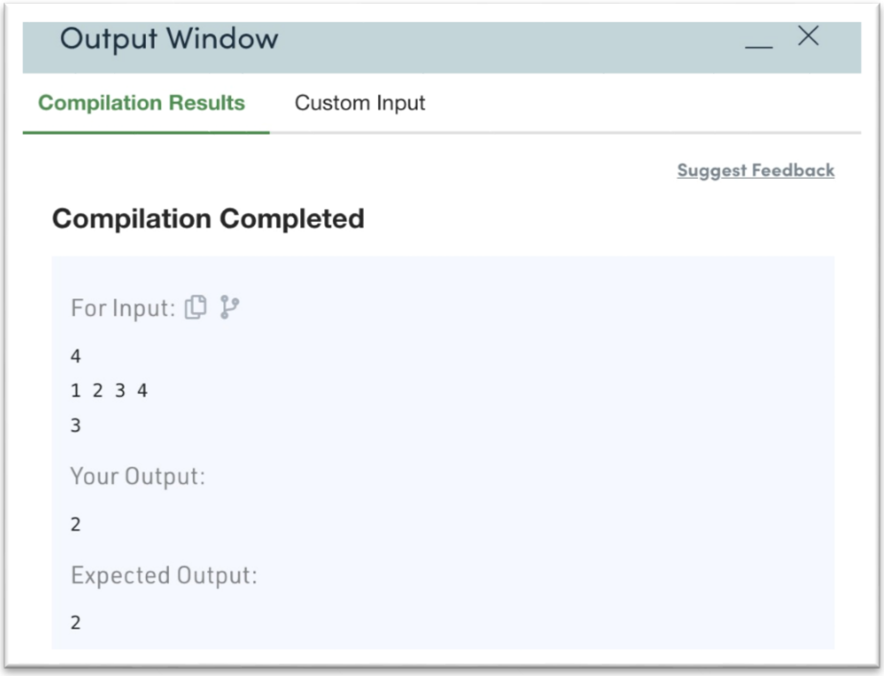
Expected Output:

```
abdes
```

Name :Jagrati Garg (28)

LINEAR SEARCH

```
class Solution{

        static int search(int arr[], int N, int X)
        {

        // Your code hereint
        idx=0;
        for(int i=0;i<N;i++){
                if(arr[i]==X){ idx=i;
                break;
                }
                else{
                idx=-1;
                }
        }
        return idx;
        }
```



Compilation Completed

For Input:

```
4
1 2 3 4
3
```

Your Output:

```
2
```

Expected Output:

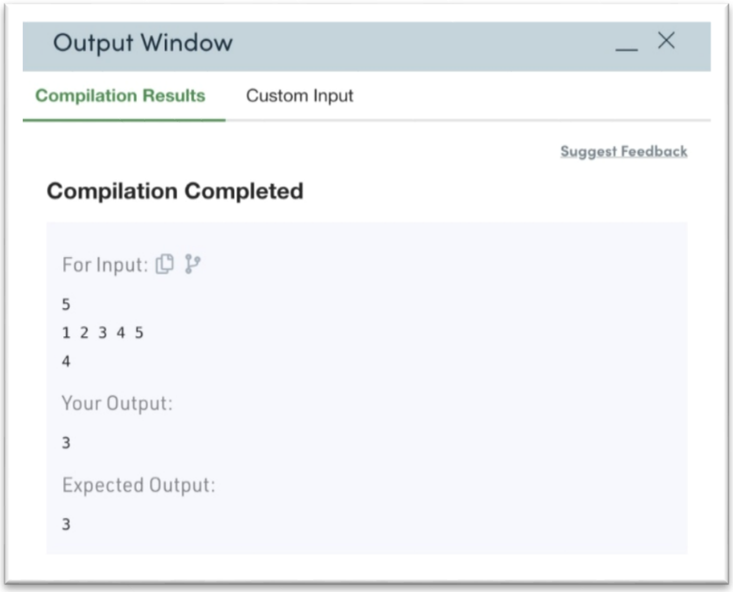
```
2
```

```
}
```

Name :Jagrati Garg (28)

BINARY SEARCH

```
class Solution
{
        int binarysearch(int arr[], int n, int k) {
        // code here int min
        = 0; int hi = n-1;
        int mid = (min+hi)/2;
        while(min<=hi){
                if(arr[mid]==k){return
                mid;
                }else if(arr[mid]<k){min =
                mid+1;
                }else{
                hi = mid-1;
                }
                mid = (min+hi)/2;
        }
        if(min>hi){
                return -1;
        }
        return mid;
        }
}
```



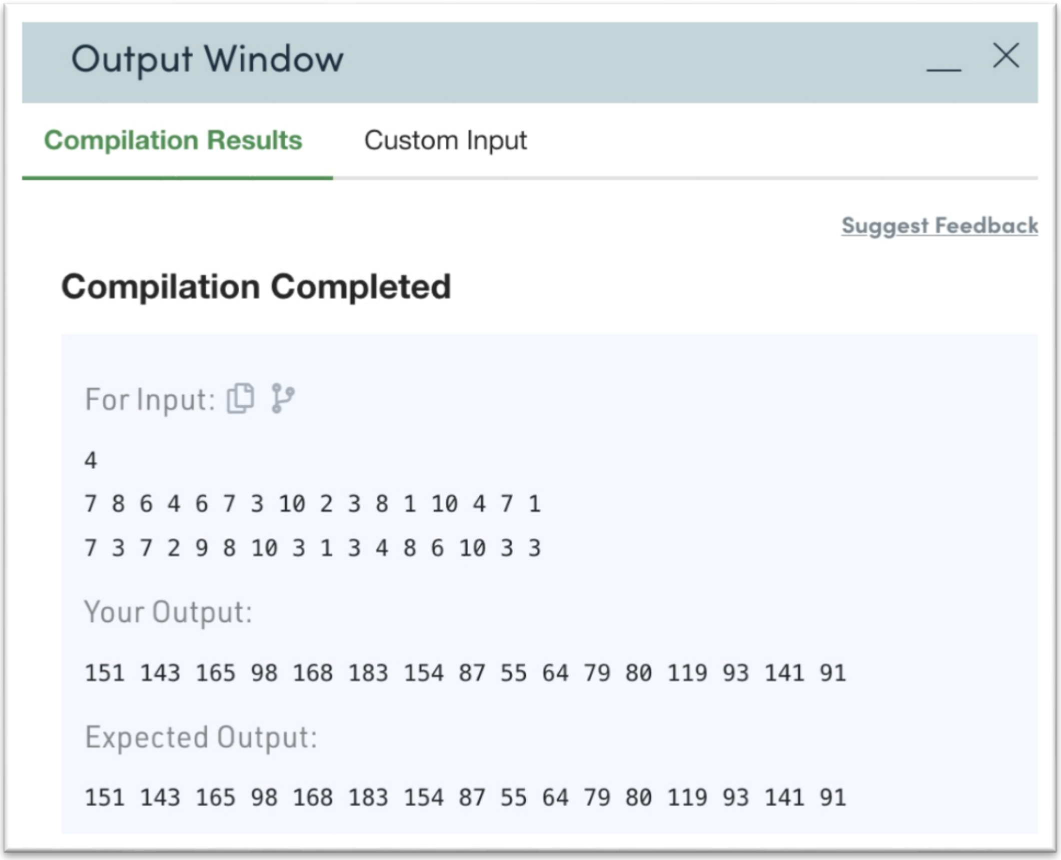
Compilation Completed

```
For Input:
5
1 2 3 4 5
4
Your Output:
3
Expected Output:
3
```

Name :Jagrati Garg (28)

MATRIX MULTIPLICATION

```
class Solution
{
        public static void multiply(int A[][], int B[][], int C[][], int N)
        {
                //add code here.
                for(int i=0;i<A.length;i++){for(int
                j=0;j<B.length;j++){
                        for (int k = 0; k < N; k++) { C[i][j] += A[i][k] *
                                B[k][j];
                                }
                }
                }
        }
}
```



Compilation Completed

For Input:

```
4
7 8 6 4 6 7 3 10 2 3 8 1 10 4 7 1
7 3 7 2 9 8 10 3 1 3 4 8 6 10 3 3
```

Your Output:

```
151 143 165 98 168 183 154 87 55 64 79 80 119 93 141 91
```

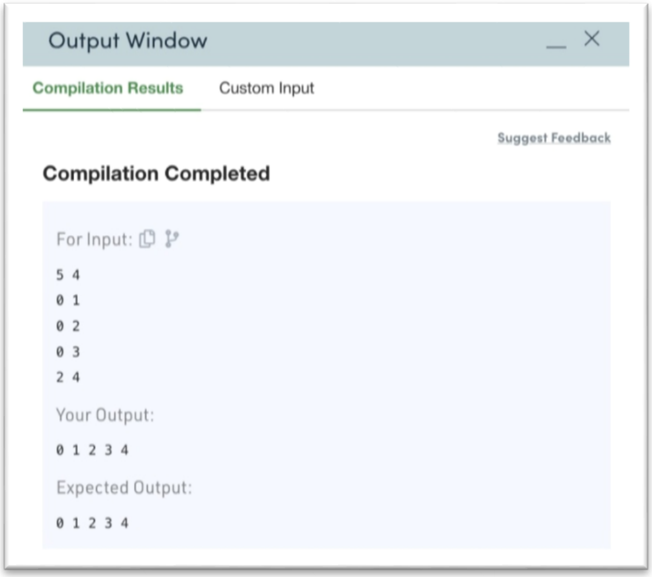
Expected Output:

```
151 143 165 98 168 183 154 87 55 64 79 80 119 93 141 91
```

Name :Jagrati Garg (28)

IMPLEMENTATION OF BREADTH FIRST SEARCH

```
class Solution {
        // Function to return Breadth First Traversal of given graph.public ArrayList<Integer>
        bfsOfGraph(int V,
ArrayList<ArrayList<Integer>> adj) { ArrayList<Integer> ans= new
        ArrayList<>();boolean visited[]= new boolean[V];
        Queue<Integer> q= new LinkedList<>(); q.add(0);

        while(!q.isEmpty())
        {
                int ele=q.remove();
                if(!visited[ele])
                {
                visited[ele]=true;
                ans.add(ele);
                for(int i=0;i<adj.get(ele).size();i++)
                {
                        q.add(adj.get(ele).get(i));
                }
                }
        }
        return ans;
        }
```



Compilation Completed

For Input:

```
5 4
0 1
0 2
0 3
2 4
```

Your Output:

```
0 1 2 3 4
```

Expected Output:

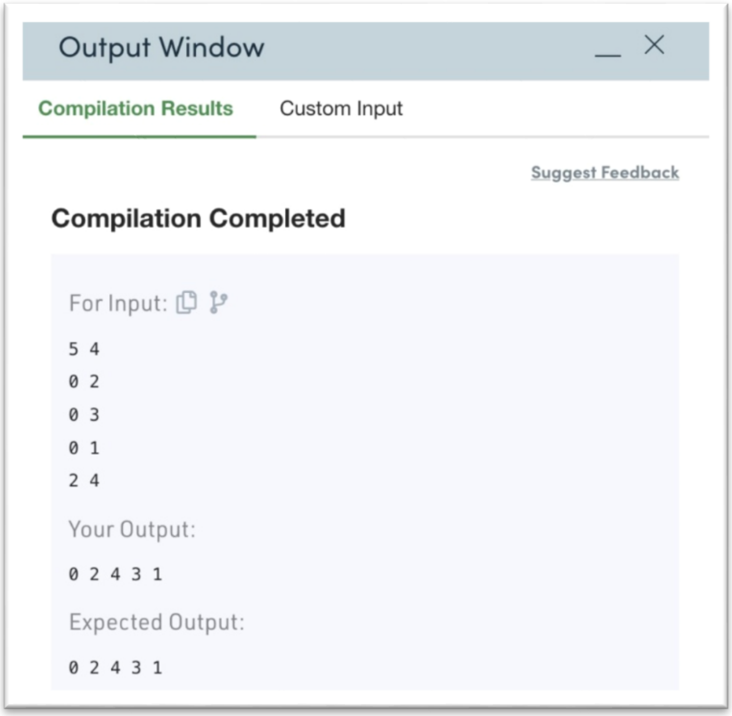
```
0 1 2 3 4
```

```
}
```

Name :Jagrati Garg (28)

IMPLEMENTATION OF DEPTH FIRST SEARCH

```
class Solution
  {
          // Function to return a list containing the DFS traversal of the graph.public ArrayList<Integer>
          dfsOfGraph(int V,
ArrayList<ArrayList<Integer>> adj) { ArrayList<Integer> arr=new
          ArrayList<>();boolean vis[]=new boolean[V];
          dfsUtil(adj,arr,vis,0);
          return arr;
          }
          public void dfsUtil(ArrayList<ArrayList<Integer>> adj,ArrayList<Integer>arr,boolean vis[],int src)
          {
          arr.add(src);
          vis[src]=true;
          for(int i=0;i<adj.get(src).size();i++){ if(!vis[adj.get(src).get(i)]){
                    dfsUtil(adj,arr,vis,adj.get(src).get(i));
                    }
          }
          }
```



Compilation Completed

For Input:

```
5 4
0 2
0 3
0 1
2 4
```

Your Output:

```
0 2 4 3 1
```

Expected Output:

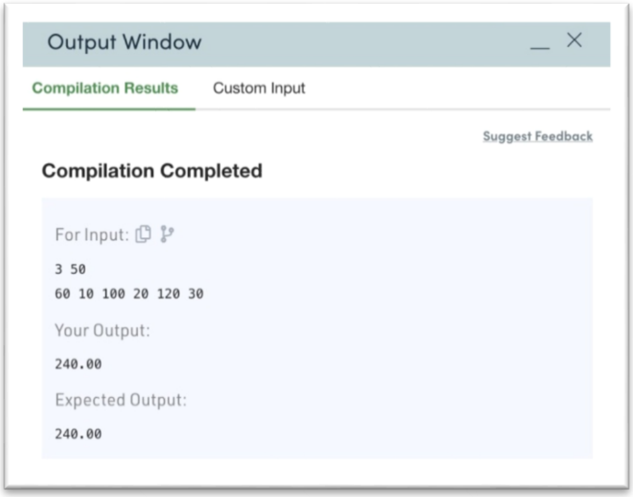
```
0 2 4 3 1
```

```
}
```

Name :Jagrati Garg (28)

FRACTIONAL KNAPSACK PROBLEM

```
class Solution
{
        //Function to get the maximum total value in the knapsack.double fractionalKnapsack(int
        W, Item arr[], int n)
        {
                // Your code here
                double ans[][]=new double[n][3];for(int
                i=0;i<n;i++){
                        ans[i][0]=arr[i].weight;
                        ans[i][1]=arr[i].value;
                        ans[i][2]=ans[i][1]/ans[i][0];
                }
                        Arrays.sort(ans, Comparator.comparingDouble(o -> -o[2]));
                double x=0; double
                curC=W;
                for(int i=0;i<ans.length;i++){
                        if(ans[i][0]<=curC){
                                  curC-=ans[i][0];
                                x+=ans[i][1];

                        }else{
                                x+=(ans[i][1]*curC/ans[i][0]);break;
                        }
                }
                return x;
        }
```



Compilation Completed

For Input:

```
3 50
60 10 100 20 120 30
```

Your Output:

```
240.00
```

Expected Output:

```
240.00
```

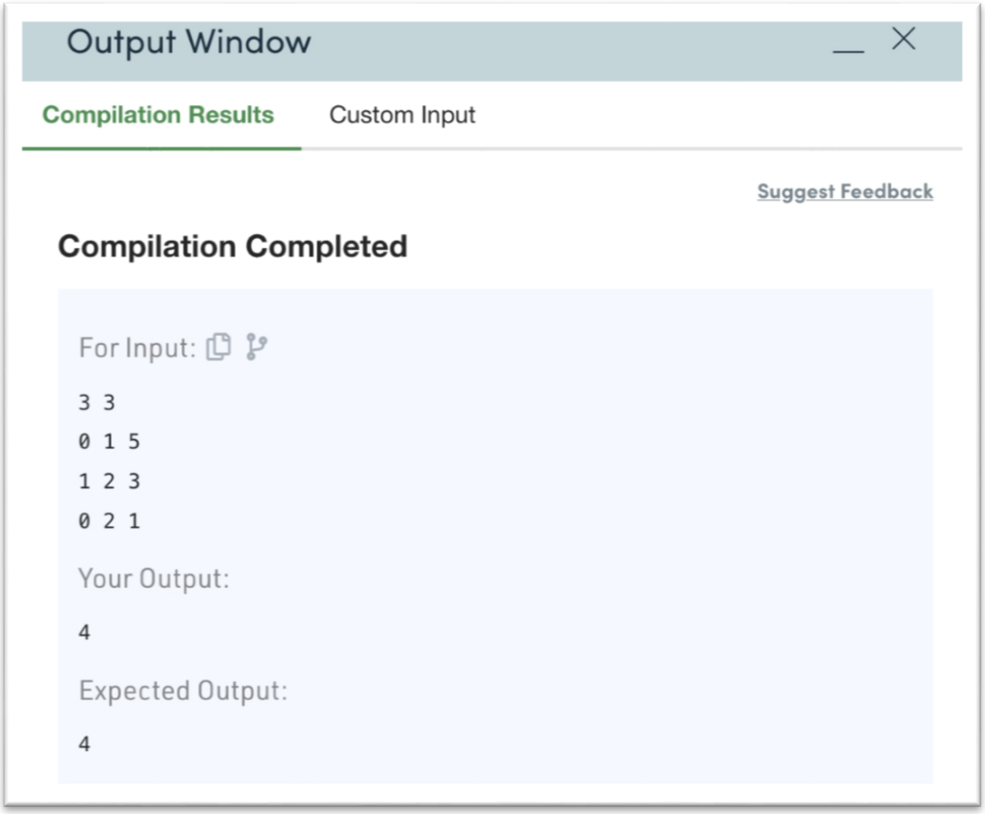
```
}
```

MINIMUM SPANNING TREES

❖ PRIM’S Algorithm

```
class  Solution{ static class
    Pair{
        int node; int
        distance;
        public Pair(int distance, int node)
        {
            this.distance=distance;this.node=node;
        }
    }
    static int spanningTree(int V, int E, int edges[][]){
        // Code Here.
        ArrayList<ArrayList<Pair>> adj=new ArrayList<>();for(int i=0;i<V;i++)
        adj.add(new ArrayList<Pair>());

        for(int i[]:edges)
        {
            int    u=i[0];    int
            v=i[1];           int
            d=i[2];
            adj.get(u).add(new Pair(d,v));
            adj.get(v).add(new Pair(d,u));
        }

        int sum=0;

        int[] vis=new int[V];



        PriorityQueue<Pair> uwu=new PriorityQueue<Pair>((x,y)-
>x.distance-y.distance);
        uwu.add(new Pair(0,0));

        while(!uwu.isEmpty())
        {
            int node=uwu.peek().node;
```

```
                        int dis=uwu.peek().distance;uwu.poll();
                        if(vis[node]==1)
                        continue;
                        sum+=dis;
                        vis[node]=1;

                        for(Pair i:adj.get(node))
                        {
                                int newnode=i.node; int
                                newdis=i.distance;

                                if(vis[newnode]==0)
                                uwu.add(new Pair(newdis, newnode));
                        }
                }
                return sum;
        }
}
```



Compilation Completed

For Input:

```
3 3
0 1 5
1 2 3
0 2 1
```

Your Output:

```
4
```

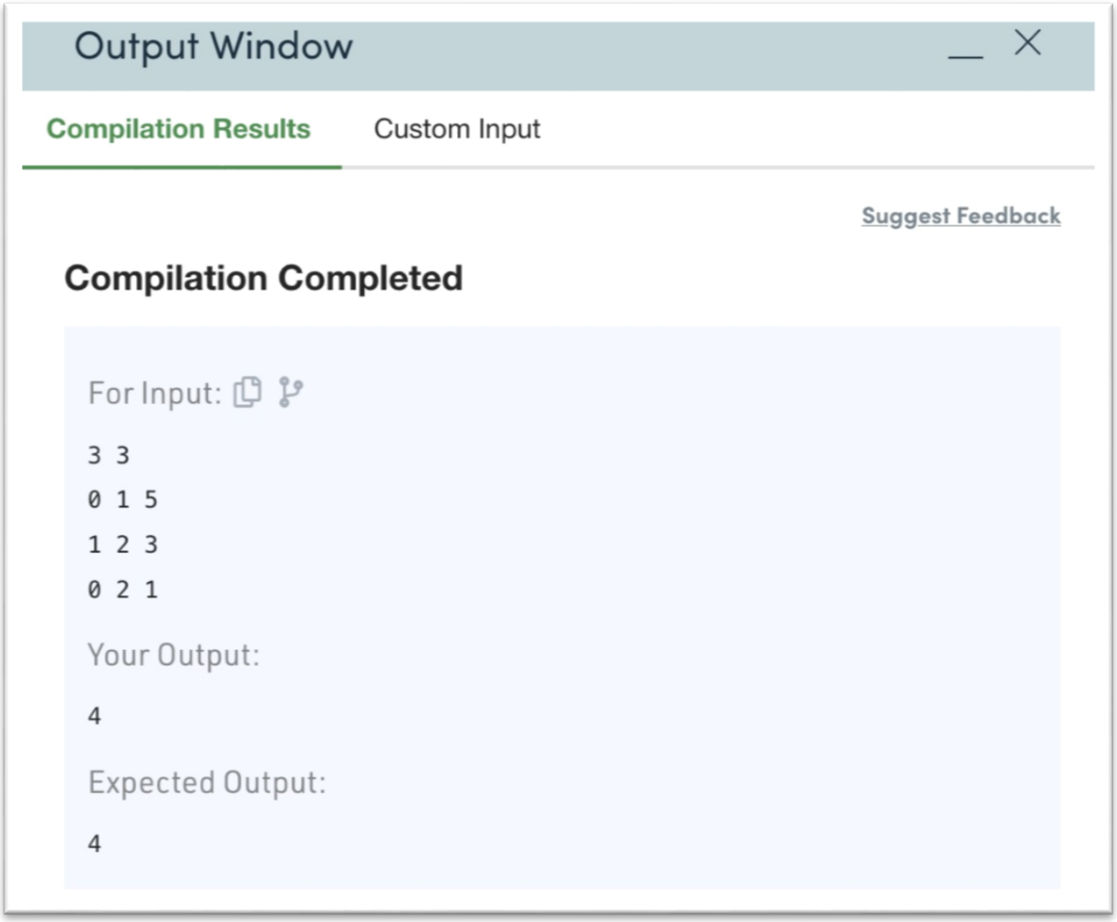
Expected Output:

```
4
```

❖ **KRUSKAL’S Algorithm**

```
class Solution{
        static int spanningTree(int V, int E, int edges[][]){int minCost = 0;
            int[] parent = new int[V+1];int[] rank =
            new int[V+1]; for(int i = 0;i<=V;i++){
                parent[i] = i;
            }
            PriorityQueue<int[]> pq = new PriorityQueue<>((a,b)->a[2]-b[2]);for(int i[] : edges){
                pq.offer(i);
            }
            while(!pq.isEmpty()){ int[] curr =
                pq.poll();
                int    p1    =    findParent(curr[0],parent); int  p2  =
                findParent(curr[1],parent);if(p1!=p2){
                    if(rank[p1]<rank[p2]){parent[p1] = p2;
                    }else if(rank[p1]>rank[p2]){parent[p2] = p1;
                    }else{
                        parent[p1] = p2;
                        rank[p1]+=1;
                    }
                    minCost += curr[2];
                }
            }
            return minCost;
        }

        static int findParent(int x,int[] parent){int tmp = x;
             while(parent[x]!=x)x =
                 parent[x];
             parent[tmp] = x; //Path Compressionreturn x;
        }
```

Name :Jagrati Garg (28)



Compilation Completed

For Input:

```
3 3
0 1 5
1 2 3
0 2 1
```

Your Output:

```
4
```

Expected Output:

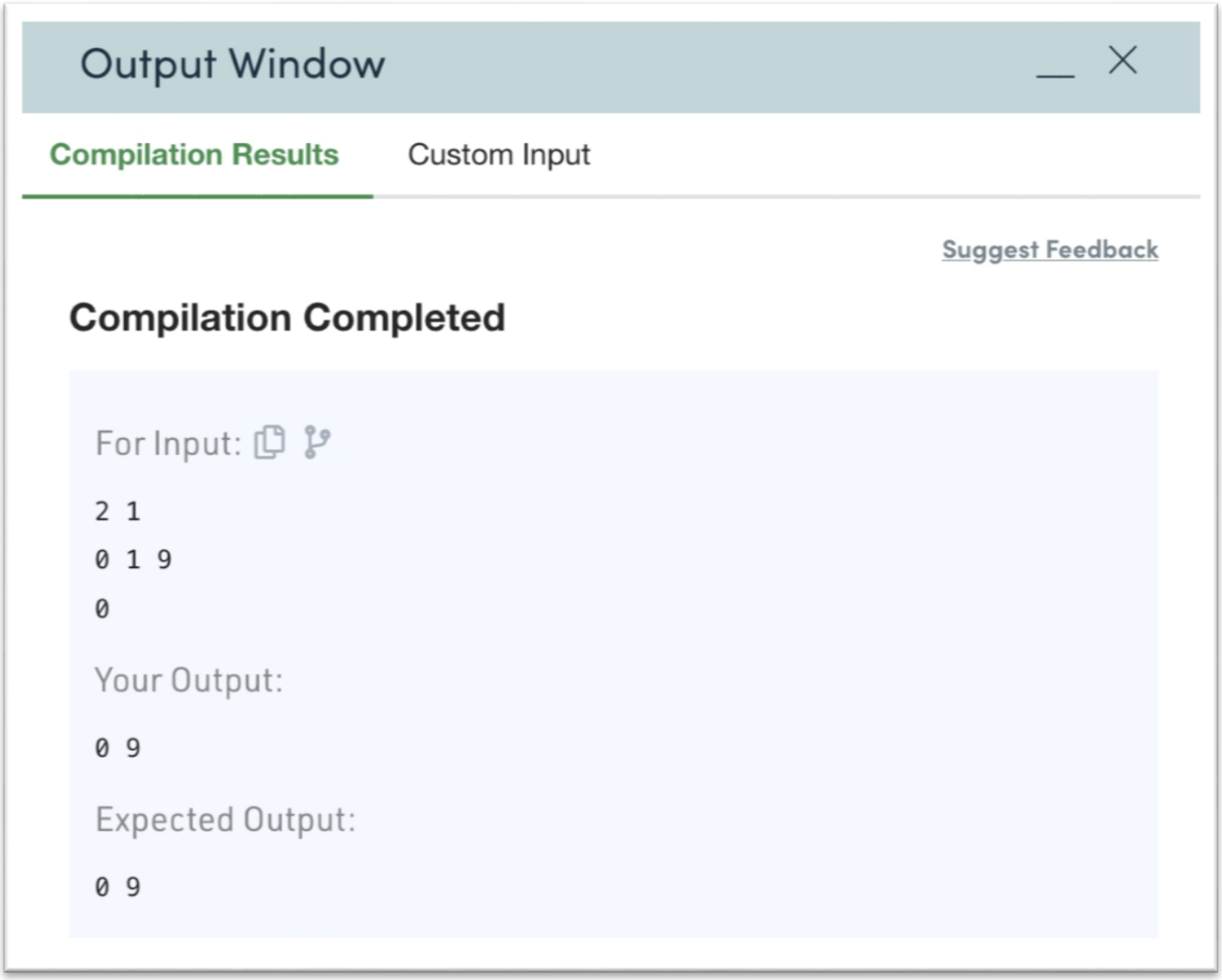
```
4
```

SINGLE SOURCE SHORTEST PATH

- **DIJKASTRA Algorithm**

```
class Solution
{
    // Solution obj=new Solution();
    //Function to find the shortest distance of all the vertices
    //from the source vertex S. static class
    Dijkastra_Pair{int vtx;
    String path;int
    cost;
    Dijkastra_Pair(int vtx,String path,int cost){this.vtx=vtx;
        this.path=path;
        this.cost=cost;
    }
    }
    static class Pair{int vtx;
    // static String path;int cost;
    Pair(int vtx,int cost){this.vtx=vtx;
        // this.path=path;
        this.cost=cost;
    }
    }
    static int[] dijkstra(int V, ArrayList<ArrayList<ArrayList<Integer>>>adj, int S)
    {
    // Write your code here
    PriorityQueue<Dijkastra_Pair> pq=new PriorityQueue<>(new Comparator<>(){public int
        compare(Dijkastra_Pair a,Dijkastra_Pair b)
        {
        return a.cost-b.cost;
        }
    });
    HashSet<Integer> visited=new HashSet<>();
    ArrayList<Pair> ans=new ArrayList<Pair>(); pq.add(new
    Dijkastra_Pair(S,String.valueOf(S),0));while(!pq.isEmpty())
    {
        Dijkastra_Pair p=pq.poll();
        if(visited.contains(p.vtx))
        {
        continue;
        }
        // System.out.println(p.vtx+" "+p.cost); ans.add(new
        Pair(p.vtx,p.cost)); visited.add(p.vtx); ArrayList<ArrayList<Integer>>
        nb=adj.get(p.vtx);for(ArrayList<Integer> n:nb)
        {
        // System.out.println("gh"); if(!visited.contains(n.get(0)))
            {
                int cost=n.get(1)+p.cost;
                // System.out.print(n.get(0)+" "+cost+":");pq.add(new
Dijkastra_Pair(n.get(0),p.path+String.valueOf(n.get(0)),cost));
            }
        }
        // System.out.println();
    }
    // Collections.sort(ans,(a,b)->a.vtx-b.vtx);int[] answer=new
    int[V];
    for(int i=0;i<ans.size();i++)
    {
    // System.out.println(ans.get(i).vtx+" "+ans.get(i).cost);

        answer[ans.get(i).vtx]=ans.get(i).cost;
    }
    return answer;
    }
}
```

Name :Jagrati Garg (28)



Compilation Completed

For Input:

```
2 1
0 1 9
0
```

Your Output:

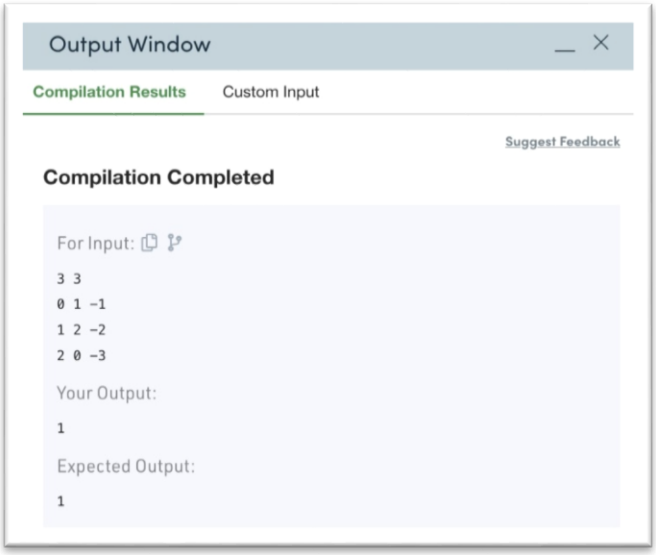
```
0 9
```

Expected Output:

```
0 9
```

- **BELLMAN FORD Algorithm**

```
class Solution
{
        public int isNegativeWeightCycle(int n, int[][] edges)
        { int [] dist = new int[n];for(int i
        =1;i<n;i++){
                dist[i] = 9999999;
        }
        for(int k =0;k<n-1;k++){
                for(int i =0;i<edges.length;i++){int u = edges[i][0];
                int v = edges[i][1]; int wt =
                edges[i][2];
                if((dist[u]+wt)<dist[v]){ dist[v] =
                        dist[u]+wt;
                }
                }
        }
        for(int i =0;i<edges.length;i++){int u =
                edges[i][0];
                int v = edges[i][1]; int wt =
                edges[i][2]; if((dist[u]+wt)<dist[v])
                return 1;
        }
        return 0;
        }
```



Compilation Completed

For Input:

```
3 3
0 1 -1
1 2 -2
2 0 -3
```

Your Output:

```
1
```

Expected Output:

```
1
```

```
}
```

IMPLEMENTATION OF DYNAMIC PROGRAMMING

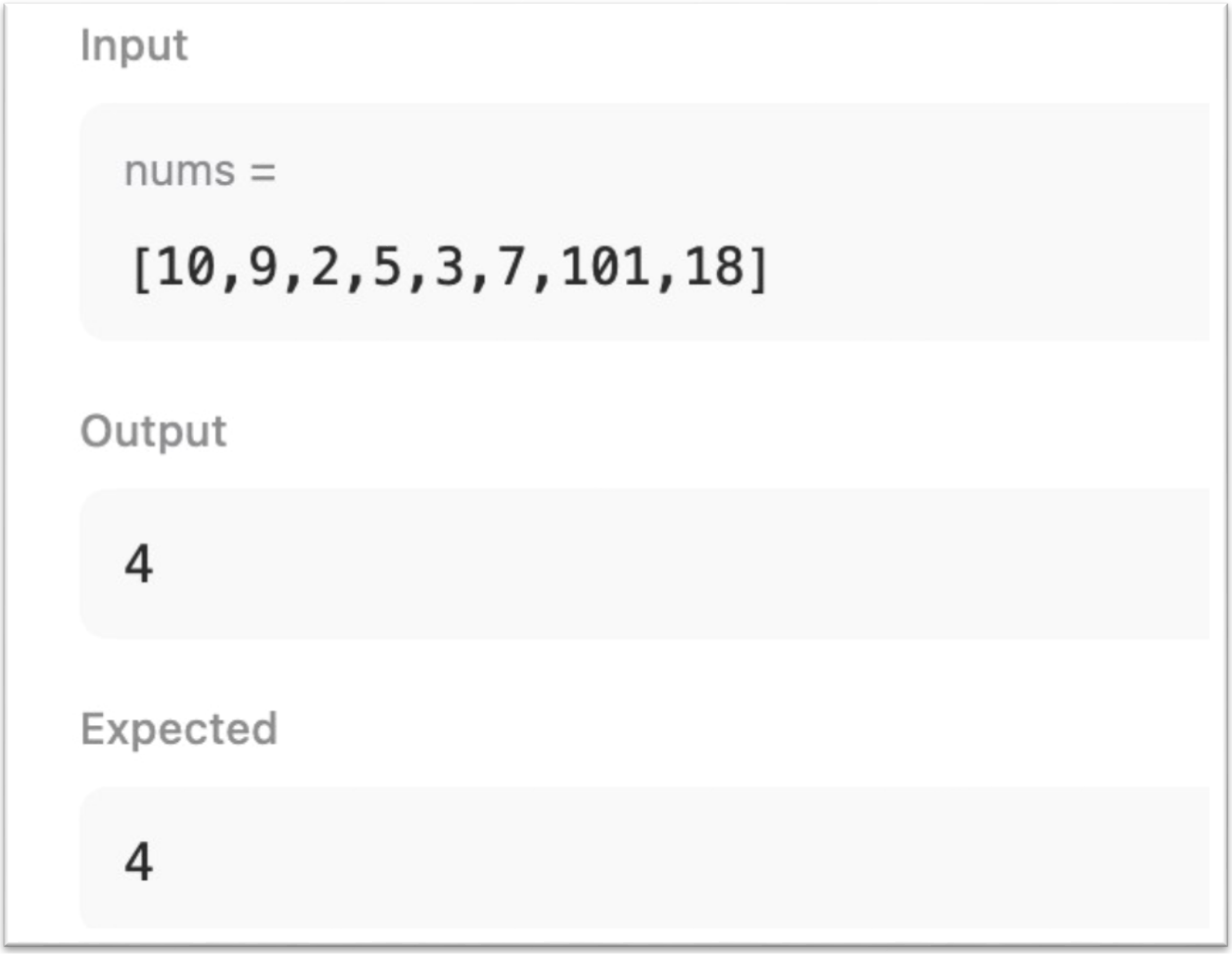
- **LONGEST INCREASING SEQUENCE**

USING RECURSION

```
class Solution {
    public int lengthOfLIS(int[] nums) { ArrayList<Integer> list=new
        ArrayList<>();return length(nums,0,-1);
    }
    public int length(int[] nums,int i,int prev){if(i==nums.length){
            return 0;
        }
        int a=length(nums,i+1,prev);//not takeint b=0;
        if(prev==-1 || nums[i]>nums[prev])
        b=1+length(nums,i+1,i);//take return
        Math.max(a,b);
    }
}
```

USING DP

```
class Solution {
    public int lengthOfLIS(int[] nums) {
        // ArrayList<Integer> list=new ArrayList<>(); int dp[][]=new
        int[nums.length][nums.length+1];for(int[] rows:dp)
            Arrays.fill(rows,-1); return
        length(nums,0,-1,dp);
    }
    public int length(int[] nums,int i,int prev,int dp[][]){if(i==nums.length)
            return 0;
        if(dp[i][prev+1]!=-1)returndp[i][prev+1];int
        a=length(nums,i+1,prev,dp);//not take int b=0;
        if(prev==-1 || nums[i]>nums[prev])
        b=1+length(nums,i+1,i,dp);//take return
        dp[i][prev+1]=Math.max(a,b);
    }
}
```



Input

```
nums =
[10,9,2,5,3,7,101,18]
```

Output

```
4
```

Expected

```
4
```

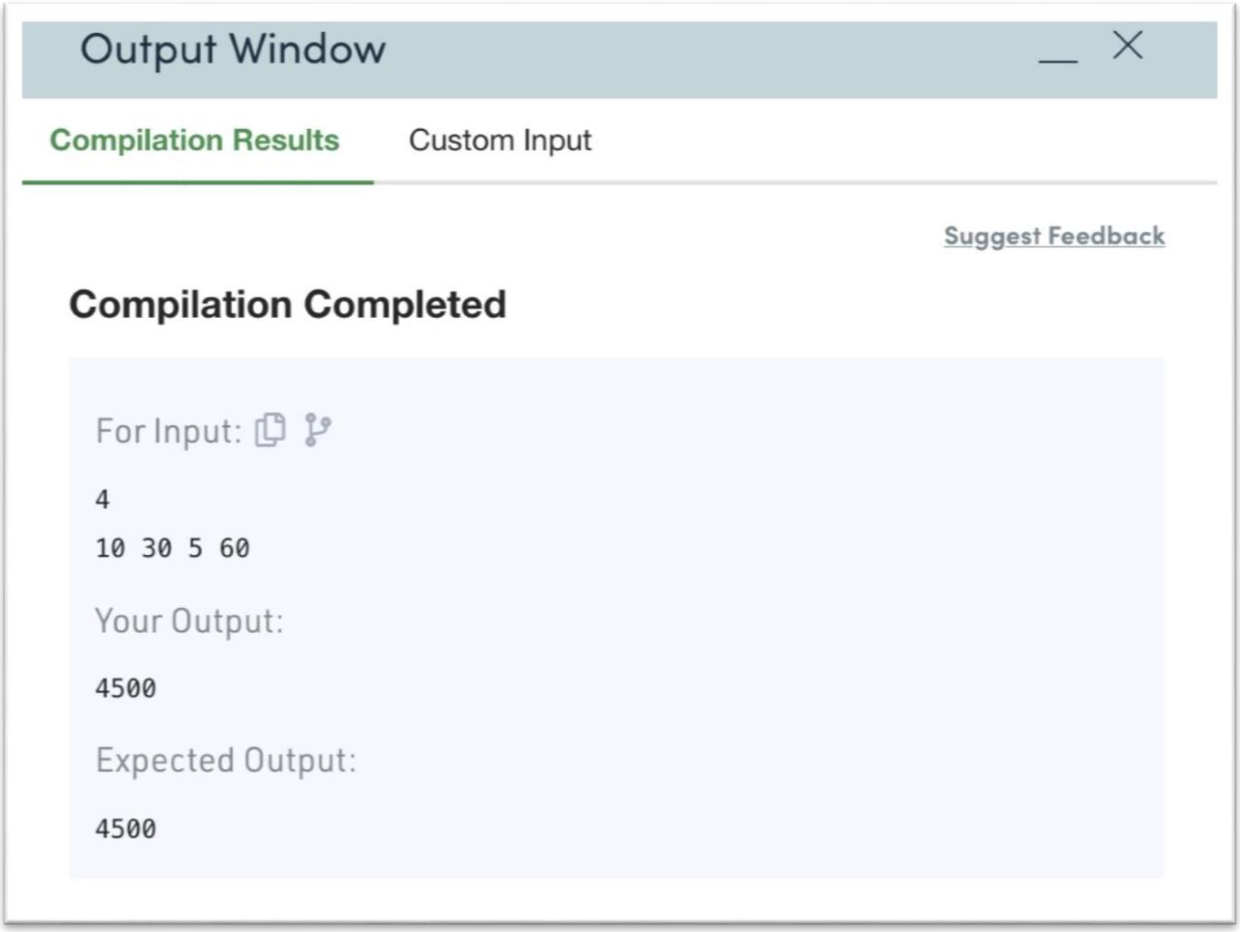
❖ MATRIX CHAIN MULTIPLICATION

USING RECURSION

```
class Solution{
        static int matrixMultiplication(int N, int arr[])
        {
                // code here
                returnmatrixMultiplication(arr,1,N-1);
        }
        public static int matrixMultiplication(int[] arr,int i,int j){if(i==j)return 0;
                int min=Integer.MAX_VALUE;for(int
                k=i;k<=j-1;k++){
                        int steps=arr[i-1]*arr[k]*arr[j]+ matrixMultiplication(arr,i,k)+matrixMultiplication(arr,k+1,j);
                        min=Math.min(min,steps);
                }
                return min;
        }
}
```

USING DP

```
class Solution{
            static int matrixMultiplication(int N, int arr[])
            {
            // code here
            int dp[][]=new int[N][N];
            for(int[] rows:dp)Arrays.fill(rows,-1); return
            matrixMultiplication(arr,1,N-1,dp);
            }
            public static int matrixMultiplication(int[] arr,int i,int j,int[][]
dp){
            if(i==j)return 0;
            if(dp[i][j]!=-1)return dp[i][j];int
            min=Integer.MAX_VALUE; for(int k=i;k<=j-
            1;k++){
                        int steps=arr[i-1]*arr[k]*arr[j]+
matrixMultiplication(arr,i,k,dp)+
                    matrixMultiplication(arr,k+1,j,dp);min=Math.min(min,steps);
            }
            return dp[i][j]=min;
            }
    }
```



Compilation Completed

For Input:

```
4
10 30 5 60
```

Your Output:

```
4500
```

Expected Output:

```
4500
```

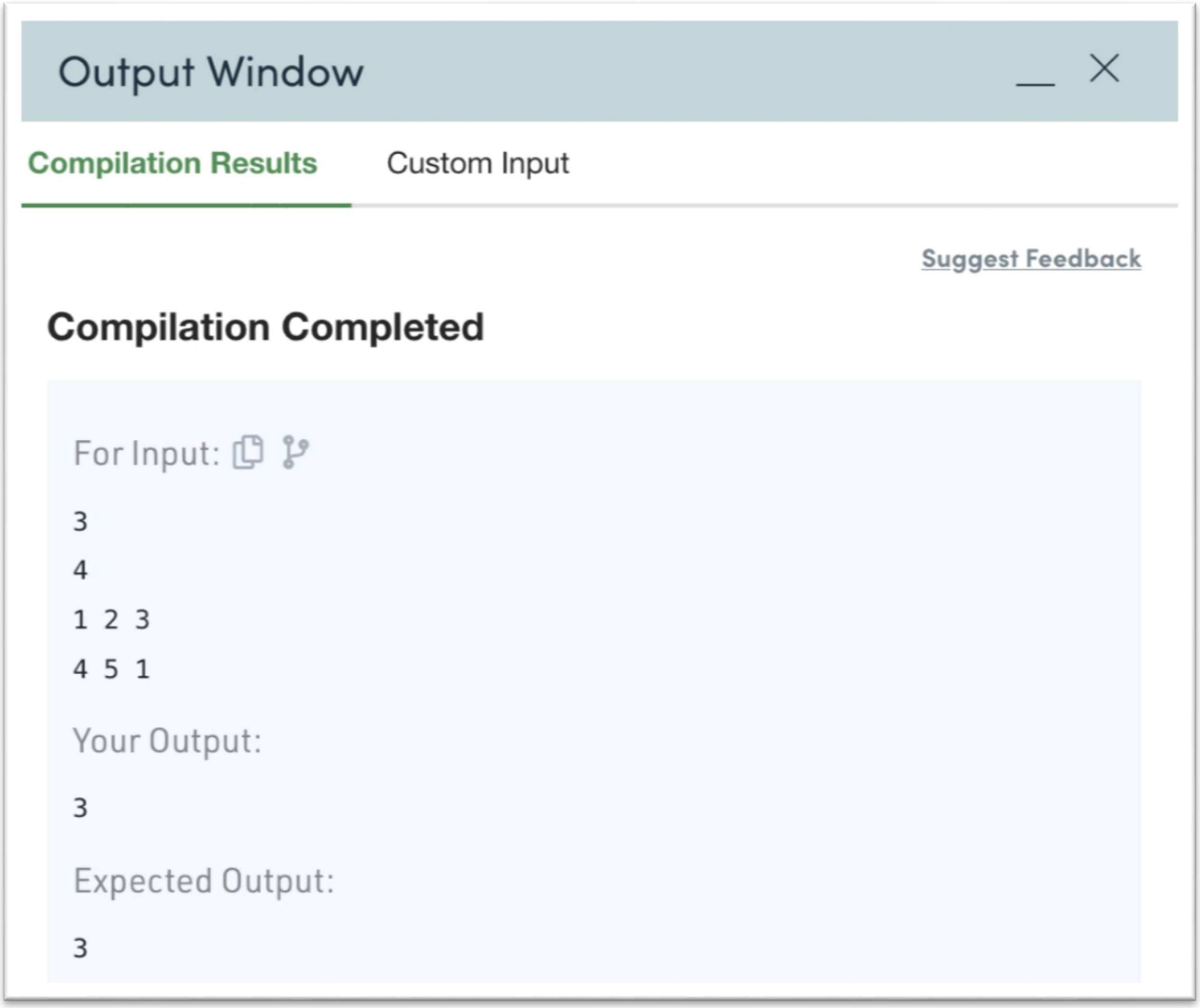
❖ 0/1 KNAPSACK PROBLEM

USING RECURSION

```
class Solution
{
        static int knapSack(int W, int wt[], int val[], int n)
        {
        // your code here
        return total(wt,val,W,n-1);
        }
        public static int total(int[] wt,int[] val,int W,int i){if(i==0){
                if(wt[0]<=W)return val[0];else return
                0;
        }
        intnottake=0+total(wt,val,W,i-1);int take=0;
        if(wt[i]<=W){
                take=val[i]+total(wt,val,W-wt[i],i-1);
        }
        return Math.max(take,nottake);
        }
}
```

USING DP

```
class Solution
{


        static int knapSack(int W, int wt[], int val[], int n)
        {
        // your code here
        int dp[][]=new int[n+1][W+1];for(int[]
        rows:dp) Arrays.fill(rows,-1);
        return total(wt,val,W,n-1,dp);
        }
        public static int total(int[] wt,int[] val,int W,int i,int[][] dp){if(i==0){
                if(wt[0]<=W)return val[0];
                else return 0;
        }
        if(dp[i][W]!=-1)return dp[i][W];
        intnottake=0+total(wt,val,W,i-1,dp);int take=0;
        if(wt[i]<=W){
                take=val[i]+total(wt,val,W-wt[i],i-1,dp);
        }
        return dp[i][W]=Math.max(take,nottake);
        }
}
```



Compilation Completed

```
For Input:

3
4
1 2 3
4 5 1

Your Output:

3

Expected Output:

3
```