COMPETITIVE PROGRAMMING

ASSIGNMENT-2

Name: V. Vyshnavi

Hallticket No.:2303A51968

Batch-14

Problem Statement :

A warehouse records package priority scores as integers. To dispatch efficiently, you must sort the scores in non-decreasing order using merge sort (divide and conquer). For each test case, output the sorted list.

Input Format:

The first line contains integer T.

For each test case:

- First line: N

- Second line: N integers (priority scores)

Output Format:

For each test case, print the sorted array in one line (space-separated).

Constraints:

- 1 ≤ T ≤ 20

- 1 ≤ N ≤ 200000 (sum of N over all test cases ≤ 200000)

- -10^9 ≤ Ai ≤ 10^9

Sample Input

1

7

4 1 6 2 5 3 2

Expected Output

1 2 2 3 4 5 6

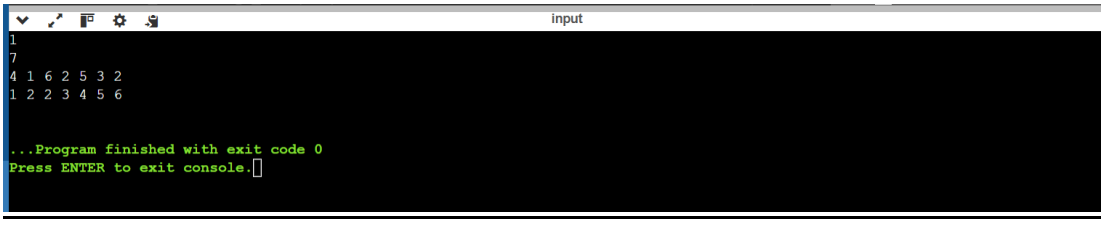
Python Code:

```
def merge_sort(arr):
    if len(arr) <= 1:
```

```
        return arr

    mid = len(arr) // 2
    left = merge_sort(arr[:mid])
    right = merge_sort(arr[mid:])

    return merge(left, right)


def merge(left, right):
    result = []
    i = j = 0

    while i < len(left) and j < len(right):
        if left[i] <= right[j]:
            result.append(left[i])
            i += 1
        else:
            result.append(right[j])
            j += 1

    result.extend(left[i:])
    result.extend(right[j:])
    return result


T = int(input())

for _ in range(T):
    N = int(input())
    arr = list(map(int, input().split()))
```

```
sorted_arr = merge_sort(arr)

print(*sorted_arr)
```

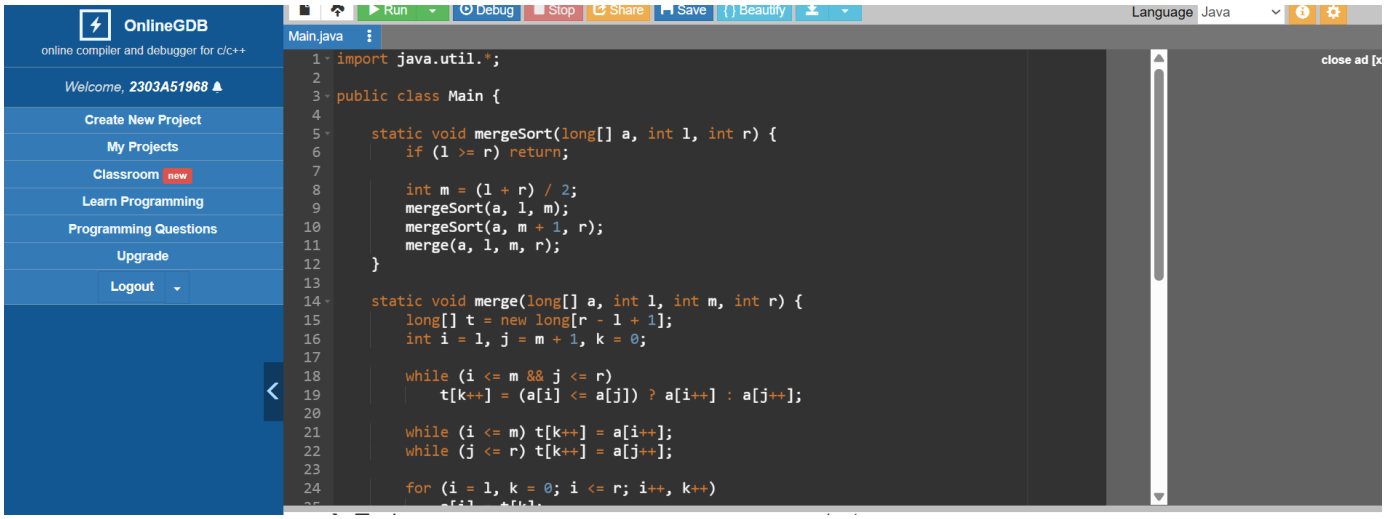
Output:



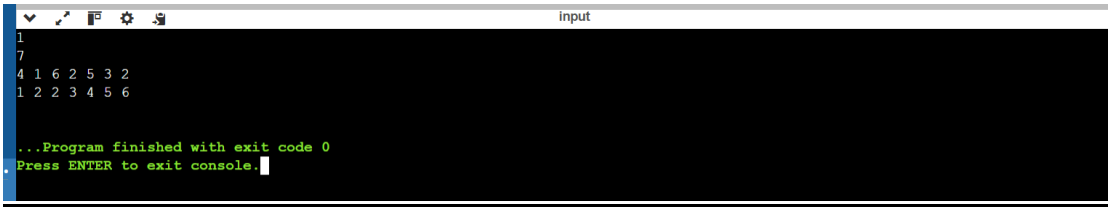
Java Code:

```
import java.util.*;

public class Main {

   static void mergeSort(long[] a, int l, int r) {
      if (l >= r) return;

      int m = (l + r) / 2;
      mergeSort(a, l, m);
      mergeSort(a, m + 1, r);
      merge(a, l, m, r);
   }

   static void merge(long[] a, int l, int m, int r) {
      long[] t = new long[r - l + 1];
      int i = l, j = m + 1, k = 0;

      while (i <= m && j <= r)
         t[k++] = (a[i] <= a[j]) ? a[i++] : a[j++];

      while (i <= m) t[k++] = a[i++];
      while (j <= r) t[k++] = a[j++];

      for (i = l, k = 0; i <= r; i++, k++)
         a[i] = t[k];
   }

   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int T = sc.nextInt();

      while (T-- > 0) {
```

```
        int N = sc.nextInt();
        long[] a = new long[N];

        for (int i = 0; i < N; i++)
            a[i] = sc.nextLong();

        mergeSort(a, 0, N - 1);

        for (int i = 0; i < N; i++)
            System.out.print(a[i] + " ");
        System.out.println();
    }
    sc.close();
  }
}
```



Output:



Maximum Profit Streak (Divide and Conquer)

Problem Statement:

A company tracks daily profit changes as an array A (values can be negative). A “profit streak” is any non-empty contiguous subarray. Find the maximum possible sum of a profit streak using a divide-and-conquer approach (split into left, right, and crossing subproblems).

Input Format:

The first line contains integer T. For each test case:

- First line: N

- Second line: N integers A1.

AN Output Format: For each test case, print one integer: maximum subarray sum.

Constraints

- 1 ≤ T ≤ 20

- 1 ≤ N ≤ 200000 (sum of N over all test cases ≤ 200000)

– -10^9 ≤ Ai ≤ 10^9

Sample Input

1

9

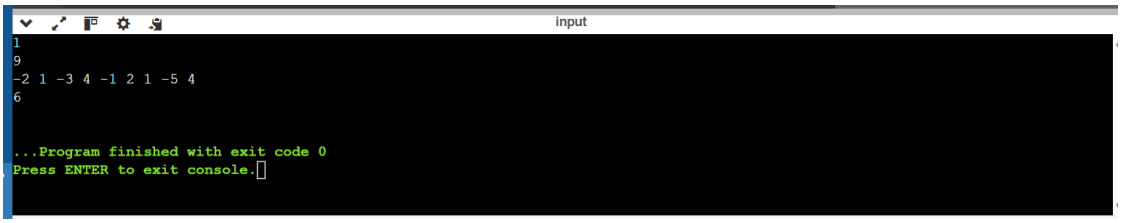
-2 1 -3 4 -1 2 1 -5 4

Expected Output : 6

Python code:

```
def max_sum(arr, l, r):
    if l == r: return arr[l]
    m = (l + r) // 2
    left = max_sum(arr, l, m)
    right = max_sum(arr, m+1, r)
    cross = sum_ = 0
    c = -10**18
    for i in range(m, l-1, -1):
        sum_ += arr[i]
        c = max(c, sum_)
    sum_ = 0
    d = -10**18
    for i in range(m+1, r+1):
        sum_ += arr[i]
        d = max(d, sum_)
    return max(left, right, c+d)

T = int(input())
for _ in range(T):
    N = int(input())
    arr = list(map(int, input().split()))
    print(max_sum(arr, 0, N-1))
```

Output:



Java Code:

```
import java.util.*;
public class Main {
   static long maxSum(long[] a, int l, int r) {
      if(l==r) return a[l];
      int m=(l+r)/2;
      long left=maxSum(a,l,m), right=maxSum(a,m+1,r);
      long sum=0,c=-Long.MAX_VALUE;
      for(int i=m;i>=l;i--){sum+=a[i]; c=Math.max(c,sum);}
      sum=0; long d=-Long.MAX_VALUE;
      for(int i=m+1;i<=r;i++){sum+=a[i]; d=Math.max(d,sum);}
      return Math.max(Math.max(left,right),c+d);
   }
   public static void main(String[] args){
      Scanner sc=new Scanner(System.in);
      int T=sc.nextInt();
```

```
        while(T-->0){
            int N=sc.nextInt();
            long[] a=new long[N];
            for(int i=0;i<N;i++) a[i]=sc.nextLong();
            System.out.println(maxSum(a,0,N-1));
        }
        sc.close();
    }
}
```

Output: