# Rajalakshmi Engineering College

Name: Pranav Shanmugam
Email: 240701395@rajalakshmi.edu.in
Roll no: 2116240701395
Phone: 8925357178
Branch: REC
Department: I CSE FD
Batch: 2028
Degree: B.E - CSE

Scan to verify results

## NeoColab_REC_CS23231_DATA STRUCTURES

### REC_DS using C_Week 6_CY_Updated

Attempt : 1
Total Mark : 30
Marks Obtained : 30

### Section 1 : Coding

1. Problem Statement

Ravi is given an array of integers and is tasked with sorting it in a unique way. He needs to sort the elements in such a way that the elements at odd positions are in descending order, and the elements at even positions are in ascending order. Ravi decided to use the Insertion Sort algorithm for this task.

Your task is to help ravi, to create even_odd_insertion_sort function to sort the array as per the specified conditions and then print the sorted array.

Example

Input:
10

25 36 96 58 74 14 35 15 75 95

Output:

96 14 75 15 74 36 35 58 25 95

***Input Format***

The first line of input consists of a single integer, N, which represents the size of the array.

The second line contains N space-separated integers, representing the elements of the array.

***Output Format***

The output displays the sorted array using the even-odd insertion sort algorithm and prints the sorted array.

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: 4
3 1 4 2
Output: 4 1 3 2

***Answer***

```
#include <stdio.h>

void even_odd_insertion_sort(int arr[], int n) {
   for (int i = 2; i < n; i++) {
      int key = arr[i];
      int j = i - 2;
      if (i % 2 == 1) {
         while (j >= 0 && arr[j] > key) {
            arr[j + 2] = arr[j];
            j -= 2;
         }
      } else {
         while (j >= 0 && arr[j] < key) {
```

```
            arr[j + 2] = arr[j];
            j -= 2;
        }
    }
    arr[j + 2] = key;
  }
}

int main() {
  int n;
  scanf("%d", &n);
  int arr[n];
  for (int i = 0; i < n; i++) {
    scanf("%d", &arr[i]);
  }
  even_odd_insertion_sort(arr, n);
  for (int i = 0; i < n; i++) {
    printf("%d ", arr[i]);
  }
  printf("\n");
  return 0;
}
```

***Status :*** Correct **Marks : 10/10**

2. Problem Statement

Marie, the teacher, wants her students to implement the ascending order of numbers while also exploring the concept of prime numbers.

Students need to write a program that sorts an array of integers using the merge sort algorithm while counting and returning the number of prime integers in the array. Help them to complete the program.

***Input Format***

The first line of input consists of an integer N, representing the number of array elements.

The second line consists of N space-separated integers, representing the array elements.

***Output Format***

The first line of output prints the sorted array of integers in ascending order.

The second line prints the number of prime integers in the array.

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: 7
5 3 6 8 9 7 4
Output: Sorted array: 3 4 5 6 7 8 9
Number of prime integers: 3

***Answer***

```
#include <stdio.h>

void merge(int arr[], int l, int m, int r) {
    int i, j, k;
    int n1 = m - l + 1;
    int n2 = r - m;
    int L[n1], R[n2];
    for (i = 0; i < n1; i++)
        L[i] = arr[l + i];
    for (j = 0; j < n2; j++)
        R[j] = arr[m + 1 + j];
    i = 0;
    j = 0;
    k = l;
    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            arr[k] = L[i];
            i++;
        } else {
            arr[k] = R[j];
            j++;
        }
        k++;
    }
```

```
    while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;
    }
    while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;
    }
}

void mergeSort(int arr[], int l, int r) {
    if (l < r) {
        int m = l + (r - l) / 2;
        mergeSort(arr, l, m);
        mergeSort(arr, m + 1, r);
        merge(arr, l, m, r);
    }
}

int isPrime(int num) {
    if (num <= 1) return 0;
    for (int i = 2; i * i <= num; i++) {
        if (num % i == 0) return 0;
    }
    return 1;
}

int countPrimes(int arr[], int n) {
    int count = 0;
    for (int i = 0; i < n; i++) {
        if (isPrime(arr[i])) count++;
    }
    return count;
}

int main() {
    int N;
    scanf("%d", &N);
    int arr[N];
    for (int i = 0; i < N; i++)
```

```
        scanf("%d", &arr[i]);
    mergeSort(arr, 0, N - 1);
    printf("Sorted array: ");
    for (int i = 0; i < N; i++)
        printf("%d ", arr[i]);
    printf("\n");
    int primeCount = countPrimes(arr, N);
    printf("Number of prime integers: %d\n", primeCount);
    return 0;
}
```

***Status :*** Correct ***Marks : 10/10***

3. Problem Statement

Priya, a data analyst, is working on a dataset of integers. She needs to find the maximum difference between two successive elements in the sorted version of the dataset. The dataset may contain a large number of integers, so Priya decides to use QuickSort to sort the array before finding the difference. Can you help Priya solve this efficiently?

***Input Format***

The first line of input consists of an integer n, representing the size of the array.

The second line consists of n space-separated integers, representing the elements of the array.

***Output Format***

The output prints a single integer, representing the maximum difference between two successive elements in the sorted form of the array.

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: 1
10
Output: Maximum gap: 0

***Answer***

```
#include <stdio.h>

void swap(int* a, int* b) {
    int t = *a;
    *a = *b;
    *b = t;
}

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

void quickSort(int arr[], int low, int high) {
    if (low < high) {
        int pi = partition(arr, low, high);
        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}

int maxGap(int arr[], int n) {
    if (n < 2) return 0;
    int max_diff = 0;
    for (int i = 1; i < n; i++) {
        int diff = arr[i] - arr[i - 1];
        if (diff > max_diff) {
            max_diff = diff;
        }
    }
    return max_diff;
}
```

```
int main() {
    int n;
    scanf("%d", &n);
    int arr[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    quickSort(arr, 0, n - 1);
    int result = maxGap(arr, n);
    printf("Maximum gap: %d\n", result);
    return 0;
}
```

***Status :*** Correct ***Marks : 10/10***