Rajalakshmi Engineering College

Name: NITHYASHREE K
Email: 240701369@rajalakshmi.edu.in
Roll no: 240701369
Phone: 9043544115
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

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

```
// You are using GCC
#include <stdio.h>

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

int partition(int arr[], int low, int high) {
    int pivot = arr[high];
    int i = low - 1;

    for(int j = low; j < high; 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 findMaxGap(int arr[], int n) {
    if(n < 2)
        return 0;
    quickSort(arr, 0, n - 1);
    int maxGap = 0;
    for(int i = 1; i < n; i++) {
        int gap = arr[i] - arr[i - 1];
        if(gap > maxGap)
            maxGap = gap;
    }
    return maxGap;
}

int main() {
    int n;
    scanf("%d", &n);
    int arr[20];

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

    int maxGap = findMaxGap(arr, n);
    printf("Maximum gap: %d\n", maxGap);
    return 0;
}
```

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

2. Problem Statement

Ravi is given an array of integers and is tasked with sorting it uniquely. 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.

Your task is to help Ravi create a program that uses insertion sort to sort the array as per the specified conditions and then print the sorted array. Position starts from 1.

Example

Input:

Size of the array = 10

Array elements = 25 36 96 58 74 14 35 15 75 95

Output:

Resultant array = 96 14 75 15 74 36 35 58 25 95

Explanation:

Initial Array: 25 36 96 58 74 14 35 15 75 95

Elements at odd positions (1, 3, 5, 7, 9): 25 96 74 35 75

Elements at odd positions sorted descending order: 96 75 74 35 25

Elements at even positions (2, 4, 6, 8, 10): 36 58 14 15 95

Elements at even positions sorted ascending order: 14 15 36 58 95

So, the final array is 96 14 75 15 74 36 35 58 25 95.

Input Format

The first line contains an integer N, representing the number of elements in the array.

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

Output Format

The output displays integers, representing the sorted array elements separated by a space.

Refer to the sample output for formatting specifications.

Sample Test Case

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

Answer

```
// You are using GCC
#include <stdio.h>

void insertionSortDescending(int arr[], int n) {
    for (int i = 1; i < n; i++) {
        int key = arr[i];
        int j = i - 1;
        while (j >= 0 && arr[j] < key) {
            arr[j + 1] = arr[j];
            j--;
        }
        arr[j + 1] = key;
    }
}

void insertionSortAscending(int arr[], int n) {
    for (int i = 1; i < n; i++) {
        int key = arr[i];
        int j = i - 1;
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j--;
        }
        arr[j + 1] = key;
    }
}

int main() {
    int n;
    scanf("%d", &n);
    int arr[20], odd[10], even[10];
    int oddIndex = 0, evenIndex = 0;
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
        if ((i + 1) % 2 == 1) {
```

```
            odd[oddIndex++] = arr[i];
        } else {
            even[evenIndex++] = arr[i];
        }
    }
    insertionSortDescending(odd, oddIndex);
    insertionSortAscending(even, evenIndex);
    int oddPtr = 0, evenPtr = 0;
    for (int i = 0; i < n; i++) {
        if ((i + 1) % 2 == 1) {
            printf("%d ", odd[oddPtr++]);
        } else {
            printf("%d ", even[evenPtr++]);
        }
    }
    printf("\n");
    return 0;
}
```

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

3. Problem Statement

Sheela wants to distribute cookies to her children, but each child will only be happy if the cookie size meets or exceeds their individual greed factor. She has a limited number of cookies and wants to make as many children happy as possible. Priya decides to sort both the greed factors and cookie sizes using QuickSort to efficiently match cookies with children. Your task is to help Sheela determine the maximum number of children that can be made happy.

Input Format

The first line of input consists of an integer n, representing the number of children.

The second line contains n space-separated integers, where each integer represents the greed factor of a child.

The third line contains an integer m, representing the number of cookies.

The fourth line contains m space-separated integers, where each integer represents the size of a cookie.

Output Format

The output prints a single integer, representing the maximum number of children that can be made happy.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 3
1 2 3
2
1 1
Output: The child with greed factor: 1

Answer

```
#include <stdio.h>

// QuickSort implementation
void quickSort(int arr[], int low, int high) {
    if (low < high) {
        int pivot = arr[high];
        int i = low - 1, temp;
        for (int j = low; j < high; j++) {
            if (arr[j] <= pivot) {
                i++;
                temp = arr[i]; arr[i] = arr[j]; arr[j] = temp;
            }
        }
        temp = arr[i + 1]; arr[i + 1] = arr[high]; arr[high] = temp;
        int pi = i + 1;
        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}

int main() {
    int n, m;
```

```
    scanf("%d", &n);
    int greed[100];
    for (int i = 0; i < n; i++) {
        scanf("%d", &greed[i]);
    }
    scanf("%d", &m);
    int cookies[100];
    for (int i = 0; i < m; i++) {
        scanf("%d", &cookies[i]);
    }
    quickSort(greed, 0, n - 1);
    quickSort(cookies, 0, m - 1);
    int i = 0, j = 0, count = 0;
    while (i < n && j < m) {
        if (cookies[j] >= greed[i]) {
            count++;
            i++;
            j++;
        } else {
            j++;
        }
    }
    printf("The child with greed factor: %d\n", count);
    return 0;
}
```

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