**Assignment -2**

**Name:** Bhavin Patil

**Roll No.:** 66

**Class:** TY CS-D

**Assignment based on merging k sorted array**

1. **Merge Sort for K-Sorted Sub-arrays**

**Code:**

#include <bits/stdc++.h>

using namespace std;

#define N 105

void printArray(vector<int> arr, int size)

{

    for (int i = 0; i < size; i++)

    {

        cout << arr[i] << " ";

    }

}

class data

{

public:

    int val, apos, vpos;

    data(int v, int ap, int vp)

    {

        val = v;

        apos = ap;

        vpos = vp;

    }

};

struct mycomp

{

    bool operator()(data &d1, data &d2)

    {

        return d1.val > d2.val;

    }

};

vector<int> mergeSortedArrays(vector<vector<int>> arr, int K)

{

    priority\_queue<data, vector<data>, mycomp> pq;

    vector<int> ans;

    for (int i = 0; i < K; i++)

    {

        data d(arr[i][0], i, 0);

        pq.push(d);

    }

    while (!pq.empty())

    {

        data curr = pq.top();

        pq.pop();

        ans.push\_back(curr.val);

        int ap = curr.apos, vp = curr.vpos;

        if (vp + 1 < arr[ap].size())

        {

            data d(arr[ap][vp + 1], ap, vp + 1);

            pq.push(d);

        }

    }

    return ans;

}

int main()

{

    int K;

    cin >> K;

    vector<vector<int>> arr(K, vector<int>(K, 0));

    for(int i=0; i<K; i++)

    {

        for(int j=0; j<K; j++)

        {

            cin>>arr[i][j];

        }

    }

    vector<int>res = mergeSortedArrays(arr, K);

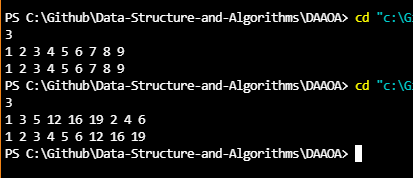
    printArray(res, K\*K);

    cout<<endl;

    return 0;

}

**Output:**



1. **Quick Sort**

**Code:**

#include <bits/stdc++.h>

using namespace std;

void printArray(int \*A, int n)

{

    for (int i = 0; i < n; i++)

        cout << " " << A[i];

    cout << endl;

}

int partition(int A[], int p, int q)

{

    int pivot = A[p];

    int i = p + 1;

    int j = q;

    int temp;

    do

    {

        while (A[i] <= pivot)

            i++;

        while (A[j] > pivot)

            j--;

        if (i < j)

        {

            temp = A[i];

            A[i] = A[j];

            A[j] = temp;

        }

    } while (i < j);

    temp = A[p];

    A[p] = A[j];

    A[j] = temp;

    return j;

}

void quickSort(int A[], int l, int h)

{

    int pivot;

    if (l < h)

    {

        pivot = partition(A, l, h);

        quickSort(A, l, pivot - 1);

        quickSort(A, pivot + 1, h);

    }

}

int main()

{

    cout << "Enter Array Length: " << endl;

    int n;

    cin >> n;

    cout << "Enter Array Elements to Sort: ";

    int A[n];

    for (int i = 0; i < n; i++)

    {

        cin >> A[i];

    }

    cout << "Unsorted Array: " << endl;

    printArray(A, n);

    quickSort(A, 0, n - 1);

    cout << "Sorted Array: " << endl;

    printArray(A, n);

    return 0;

}

// 11 13 7 12 16 9 24 5 10 3

**Output:**

