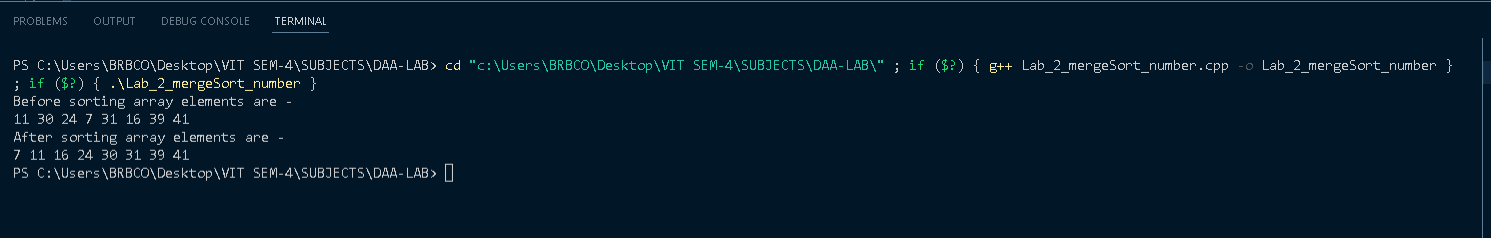
**EXERCISE-2: MERGE SORT**

**PRATEEK MOHANTY**

**20BCE1482**

* **MERGE SORT (NUMBERS)**
* #include <iostream>
* using namespace std;
* /\* Function to merge the subarrays of a[] \*/
* void merge(int a[], int beg, int mid, int end)
* {
* int i, j, k;
* int n1 = mid - beg + 1;
* int n2 = end - mid;
* int LeftArray[n1], RightArray[n2]; //temporary arrays
* /\* copy data to temp arrays \*/
* for (int i = 0; i < n1; i++)
* LeftArray[i] = a[beg + i];
* for (int j = 0; j < n2; j++)
* RightArray[j] = a[mid + 1 + j];
* i = 0;   /\* initial index of first sub-array \*/
* j = 0;   /\* initial index of second sub-array \*/
* k = beg; /\* initial index of merged sub-array \*/
* while (i < n1 && j < n2)
* {
* if (LeftArray[i] <= RightArray[j])
* {
* a[k] = LeftArray[i];
* i++;
* }
* else
* {
* a[k] = RightArray[j];
* j++;
* }
* k++;
* }
* while (i < n1)
* {
* a[k] = LeftArray[i];
* i++;
* k++;
* }
* while (j < n2)
* {
* a[k] = RightArray[j];
* j++;
* k++;
* }
* }
* void mergeSort(int a[], int beg, int end)
* {
* if (beg < end)
* {
* int mid = (beg + end) / 2;
* mergeSort(a, beg, mid);
* mergeSort(a, mid + 1, end);
* merge(a, beg, mid, end);
* }
* }
* /\* Function to print the array \*/
* void printArray(int a[], int n)
* {
* int i;
* for (i = 0; i < n; i++)
* cout << a[i] << " ";
* }
* int main()
* {
* int a[] = {11, 30, 24, 7, 31, 16, 39, 41};
* int n = sizeof(a) / sizeof(a[0]);
* cout << "Before sorting array elements are - \n";
* printArray(a, n);
* mergeSort(a, 0, n - 1);
* cout << "\nAfter sorting array elements are - \n";
* printArray(a, n);
* return 0;
* }

**OUTPUT**

****

* **MERGE SORT (CHARACTER)**

**CODE**

#include <iostream>

#include <vector>

using namespace std;

void print(vector<int> v)

{

    for (int i = 0; i < v.size(); i++)

        cout << v[i] << " ";

    cout << endl;

}

vector<char> merge(vector<char> left, vector<char> right)

{

    vector<char> result;

    int right\_side\_counter = 0, left\_side\_counter = 0;

    while ((int)left.size() > 0 || (int)right.size() > 0)

    {

        if ((int)left.size() > 0 && (int)right.size() > 0)

        {

            if ((int)left.front() <= (int)right.front())

            {

                result.push\_back((int)left.front());

                left.erase(left.begin());

            }

            else

            {

                result.push\_back((int)right.front());

                right.erase(right.begin());

            }

        }

        else if ((int)left.size() > 0)

        {

            for (int i = 0; i < (int)left.size(); i++)

                result.push\_back(left[i]);

            break;

        }

        else if ((int)right.size() > 0)

        {

            for (int i = 0; i < (int)right.size(); i++)

                result.push\_back(right[i]);

            break;

        }

    }

    return result;

}

vector<char> mergeSort(vector<char> m, int &leftCount, int &rightCount)

{

    if (m.size() <= 1)

        return m;

    vector<char> left, right, result;

    int middle = ((int)m.size() + 1) / 2;

    for (int i = 0; i < middle; i++)

    {

        left.push\_back(m[i]);

    }

    for (int i = middle; i < (int)m.size(); i++)

    {

        right.push\_back(m[i]);

    }

    leftCount++;

    left = mergeSort(left, leftCount, rightCount);

    rightCount++;

    right = mergeSort(right, leftCount, rightCount);

    result = merge(left, right);

    return result;

}

int main()

{

    int a = 0, b = 0;

    int n;

    cout << "Enter the number of elements in the vector array: \n";

    cin >> n;

    vector<char> v;

    cout << "Enter the elements of the vector array: \n";

    for (int i = 0; i < n; i++)

    {

        char x;

        cin >> x;

        v.push\_back(x);

    }

    cout << "The unsorted vector array is: \n";

    for (int i = 0; i < n; i++)

    {

        cout << v[i] << " ";

    }

    cout << "\n";

    cout << "The sorted vector array using merge sort is: \n";

    vector<char> result = mergeSort(v, a, b);

    for (int i = 0; i < n; i++)

    {

        cout << result[i] << " ";

    }

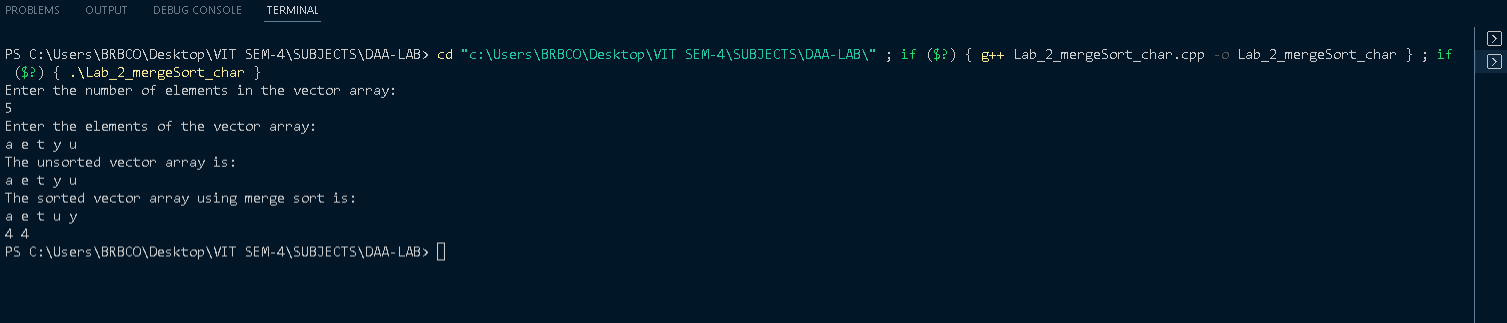
    cout << "\n";

    cout << a << " " << b;

    return 0;

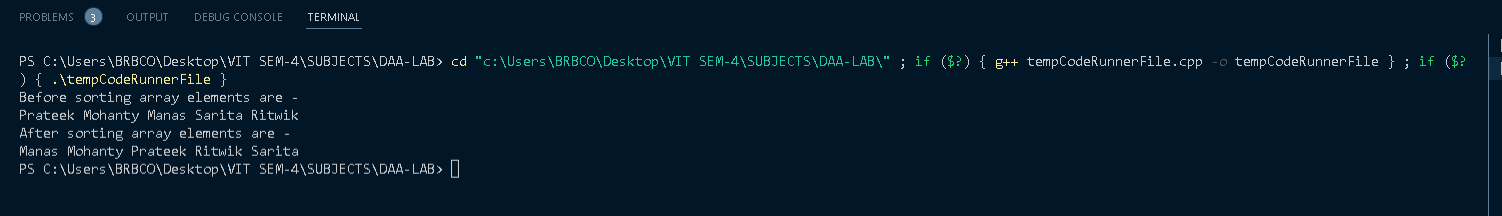
}

**OUTPUT**

****

* **MERGE SORT (STRING)**
* #include <iostream>
* using namespace std;
* /\* Function to merge the subarrays of a[] \*/
* void merge(string a[], int beg, int mid, int end)
* {
* int i, j, k;
* int n1 = mid - beg + 1;
* int n2 = end - mid;
* string LeftArray[n1], RightArray[n2]; //temporary arrays
* /\* copy data to temp arrays \*/
* for (int i = 0; i < n1; i++)
* LeftArray[i] = a[beg + i];
* for (int j = 0; j < n2; j++)
* RightArray[j] = a[mid + 1 + j];
* i = 0;   /\* initial index of first sub-array \*/
* j = 0;   /\* initial index of second sub-array \*/
* k = beg; /\* initial index of merged sub-array \*/
* while (i < n1 && j < n2)
* {
* if (LeftArray[i] <= RightArray[j])
* {
* a[k] = LeftArray[i];
* i++;
* }
* else
* {
* a[k] = RightArray[j];
* j++;
* }
* k++;
* }
* while (i < n1)
* {
* a[k] = LeftArray[i];
* i++;
* k++;
* }
* while (j < n2)
* {
* a[k] = RightArray[j];
* j++;
* k++;
* }
* }
* void mergeSort(string a[], int beg, int end)
* {
* if (beg < end)
* {
* int mid = (beg + end) / 2;
* mergeSort(a, beg, mid);
* mergeSort(a, mid + 1, end);
* merge(a, beg, mid, end);
* }
* }
* /\* Function to print the array \*/
* void printArray(string a[], int n)
* {
* int i;
* for (i = 0; i < n; i++)
* cout << a[i] << " ";
* }
* int main()
* {
* //int a[] = {11, 30, 24, 7, 31, 16, 39, 41};
* string a[] = {"Prateek", "Mohanty", "Manas", "Sarita", "Ritwik"};
* int n = sizeof(a) / sizeof(a[0]);
* cout << "Before sorting array elements are - \n";
* printArray(a, n);
* mergeSort(a, 0, n - 1);
* cout << "\nAfter sorting array elements are - \n";
* printArray(a, n);
* return 0;
* }

**OUTPUT**

****