**K23-0607 DS LAB # 7 Oct 10,2024**

**Question # 1**

#include <iostream>

using namespace std;

void swap(int &a, int &b) {

int temp = a;

a = b;

b = temp;

}

int partition\_ascending(int\* arr, int s, int e) {

    int mid = s + (e - s) / 2;

    int pivot = arr[mid];

    swap(arr[mid], arr[e]);

    int i = s - 1;

    for (int j = s; j < e; j++) {

        if (arr[j] <= pivot) {

            i++;

            swap(arr[i], arr[j]);

        }

    }

    swap(arr[i + 1], arr[e]);

    return (i + 1);

}

void Quick\_Sort\_Ascending(int\* arr, int s, int e) {

    if (s >= e) {

        return;

    }

    int p = partition\_ascending(arr, s, e);

    Quick\_Sort\_Ascending(arr, s, p - 1);

    Quick\_Sort\_Ascending(arr, p + 1, e);

}

int partition\_descending(int\* arr, int s, int e) {

    int mid = s + (e - s) / 2;

    int pivot = arr[mid];

    swap(arr[mid], arr[e]);

    int i = s - 1;

    for (int j = s; j < e; j++) {

        if (arr[j] >= pivot) {

            i++;

            swap(arr[i], arr[j]);

        }

    }

    swap(arr[i + 1], arr[e]);

    return (i + 1);

}

void Quick\_Sort\_Descending(int\* arr, int s, int e) {

    if (s >= e) {

        return;

    }

    int p = partition\_descending(arr, s, e);

    Quick\_Sort\_Descending(arr, s, p - 1);

    Quick\_Sort\_Descending(arr, p + 1, e);

}

int\* Sort\_Array\_In\_Ascending(int\* arr, int n) {

    int\* arr1 = new int[n];

    for (int i = 0; i < n; i++) {

        arr1[i] = arr[i];

    }

    Quick\_Sort\_Ascending(arr1, 0, n - 1);

    return arr1;

}

int\* Sort\_Array\_In\_Descending(int\* arr, int n) {

    int\* arr1 = new int[n];

    for (int i = 0; i < n; i++) {

        arr1[i] = arr[i];

    }

    Quick\_Sort\_Descending(arr1, 0, n - 1);

    return arr1;

}

int main() {

    int arr[10] = {9, 2, 3, 5, 7, 1, 8, 0, 6, 4};

    int n = 10;

    int\* Ascending123 = Sort\_Array\_In\_Ascending(arr, n);

    int\* Descending123 = Sort\_Array\_In\_Descending(arr, n);

    cout << "Ascending: ";

    for (int i = 0; i < n; i++) {

        cout << Ascending123[i] << " ";

    }

    cout << endl << "Descending: ";

    for (int i = 0; i < n; i++) {

        cout << Descending123[i] << " ";

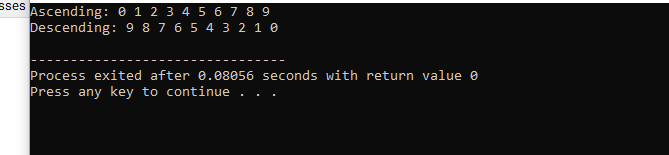
    }

    cout << endl;

    delete[] Ascending123;

    delete[] Descending123;

}



**Question # 2**

#include <iostream>

using namespace std;

void countingSort(int\* arr, int n, int exp, bool ascending) {

    int output[n];

    int count[10] = {0};

    for (int i = 0; i < n; i++) {

        count[(arr[i] / exp) % 10]++;

    }

    if (ascending) {

        for (int i = 1; i < 10; i++) {

            count[i] += count[i - 1];

        }

    }

    else {

        for (int i = 8; i >= 0; i--) {

            count[i] += count[i + 1];

        }

    }

    for (int i = n - 1; i >= 0; i--) {

        int index = (arr[i] / exp) % 10;

        output[count[index] - 1] = arr[i];

        count[index]--;

    }

    for (int i = 0; i < n; i++) {

        arr[i] = output[i];

    }

}

int\* radixSort123(int\* arr, int n, bool ascending) {

    int\* sorted = new int[n];

    for (int i = 0; i < n; i++) {

        sorted [i] = arr[i];

    }

    int maxi = sorted [0];

    for (int i = 1; i < n; i++) {

        if (sorted [i] > maxi) {

            maxi = sorted [i];

        }

    }

    for (int exp = 1; maxi / exp > 0; exp \*= 10) {

        countingSort(sorted, n, exp, ascending);

    }

    return sorted;

}

int main() {

    int arr[10] = {9, 2, 3, 5, 7, 1, 8, 0, 6, 4};

    int n = 10;

    int\* ascending = radixSort123(arr, n, true);

    int\* descending = radixSort123(arr, n, false);

    cout << "Ascending : ";

    for (int i = 0; i < n; i++) {

        cout << ascending [i] << " ";

    }

    cout << endl;

    cout << "Descending : ";

    for (int i = 0; i < n; i++) {

        cout << descending [i] << " ";

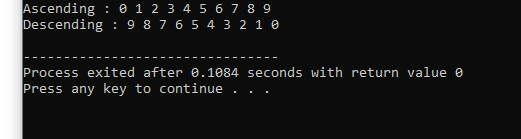
    }

    cout << endl;

    delete[] ascending;

    delete[] descending;

}



**Question # 3**

#include<iostream>

using namespace std;

void Merge\_Ascending( int\* arr, int s, int e, int mid){

    int len1 = mid - s + 1;

    int len2 = e - mid;

    int\* temp1 = new int [len1];

    int\* temp2 = new int [len2];

    int k = s;

    for( int i=0; i<len1; i++){

        temp1[i] = arr[k++];

    }

    for( int i=0; i<len2; i++){

        temp2[i] = arr[k++];

    }

    int index1 = 0;

    int index2 = 0;

    k = s;

    while( index1 < len1 && index2 < len2 ){

        if( temp1[index1] < temp2[index2] ){

            arr[k++] = temp1[index1++];

        }

        else{

            arr[k++] = temp2[index2++];

        }

    }

    while( index1 < len1 ){

        arr[k++] = temp1[index1++];

    }

    while( index2 < len2 ){

        arr[k++] = temp2[index2++];

    }

    delete[]temp1;

    delete[]temp2;

}

void Merge\_Descending( int\* arr, int s, int e, int mid){

    int len1 = mid - s + 1;

    int len2 = e - mid;

    int\* temp1 = new int [len1];

    int\* temp2 = new int [len2];

    int k = s;

    for( int i=0; i<len1; i++){

        temp1[i] = arr[k++];

    }

    for( int i=0; i<len2; i++){

        temp2[i] = arr[k++];

    }

    int index1 = 0;

    int index2 = 0;

    k = s;

    while( index1 < len1 && index2 < len2 ){

        if( temp1[index1] > temp2[index2] ){

            arr[k++] = temp1[index1++];

        }

        else{

            arr[k++] = temp2[index2++];

        }

    }

    while( index1 < len1 ){

        arr[k++] = temp1[index1++];

    }

    while( index2 < len2 ){

        arr[k++] = temp2[index2++];

    }

    delete[]temp1;

    delete[]temp2;

}

void Merge\_Sort\_Ascending( int\* arr, int s, int e){

    if( s >= e ) return;

    int mid = s + ( e-s )/2;

    Merge\_Sort\_Ascending( arr, s, mid );

    Merge\_Sort\_Ascending( arr, mid + 1, e );

    Merge\_Ascending( arr, s, e, mid );

}

void Merge\_Sort\_Descending( int\* arr, int s, int e){

    if( s >= e ) return;

    int mid = s + ( e-s )/2;

    Merge\_Sort\_Descending( arr, s, mid );

    Merge\_Sort\_Descending( arr, mid + 1, e );

    Merge\_Descending( arr, s, e, mid );

}

int\* Sort\_Array\_In\_Ascending( int\* arr, int n ){

    int \*arr1 = new int [n];

    for(int i=0; i<n; i++){

        arr1[i] = arr[i];

    }

    Merge\_Sort\_Ascending( arr1, 0, n-1 );

    return arr1;

}

int\* Sort\_Array\_In\_Descending( int\* arr, int n ){

    int \*arr1 = new int [n];

    for(int i=0; i<n; i++){

        arr1[i] = arr[i];

    }

    Merge\_Sort\_Descending( arr1, 0, n-1 );

    return arr1;

}

int main(){

    int arr[10] = {9,2,3,5,7,1,8,0,6,4};

    int n = 10;

    int\* Ascending123 = Sort\_Array\_In\_Ascending( arr, n );

    int\* Descending123 = Sort\_Array\_In\_Descending( arr, n );

    cout  << "Ascending : " << " ";

    for(int i=0; i<n; i++){

        cout << Ascending123[i] << " ";

    }

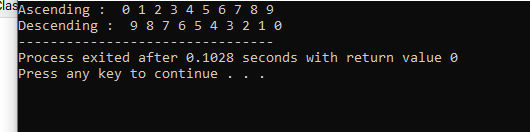
    cout << endl << "Descending : " << " ";

    for(int i=0; i<n; i++){

        cout << Descending123[i] << " ";

    }

}



**END**