**K226007**

**Yousha Mehdi**

**BSR-3C**

**LAB 7**

**Task 1:**



**#include <iostream>**

**#include <time.h>**

**#include <cstdlib>**

**using namespace std;**

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

**{**

**int temp;**

**temp = a;**

**a = b;**

**b = temp;**

**}**

**int partition(int arr[], int low, int high, int &count)**

**{**

**static bool flag = true;**

**int pivot = arr[high];**

**int i = (low - 1);**

**for (int j = low; j < high; j++)**

**{**

**if (arr[j] < pivot)**

**{**

**i++;**

**count++;**

**swap(arr[i], arr[j]);**

**}**

**}**

**swap(arr[i + 1], arr[high]);**

**return (i + 1);**

**}**

**void quickSort(int arr[], int low, int high, int &count)**

**{**

**if (low < high)**

**{**

**int pivotIndex = partition(arr, low, high, count);**

**quickSort(arr, low, pivotIndex - 1, count);**

**quickSort(arr, pivotIndex + 1, high, count);**

**}**

**}**

**int main()**

**{**

**srand(time(NULL));**

**int arr[] = {12, 11, 13, 5, 6, 7};**

**int size = sizeof(arr) / sizeof(arr[0]);**

**cout << "Unsorted array: ";**

**for (int i = 0; i < size; i++)**

**{**

**cout << arr[i] << " ";**

**}**

**cout << endl;**

**// Pivot as first element k226007**

**int count = 0;**

**quickSort(arr, 0, size-1, count);**

**cout << "\n\nSorted array: ";**

**for (int i = 0; i < size; i++)**

**{**

**cout << arr[i] << " ";**

**}**

**cout << "\nPivot first element count: " << count;**

**// Pivot as Random element k226007**

**count = 0;**

**int random = rand() % size;**

**quickSort(arr, 0, random, count);**

**cout << "\n\nSorted array: ";**

**for (int i = 0; i < size; i++)**

**{**

**cout << arr[i] << " ";**

**}**

**cout << "\nPivot Random element count: " << count;**

**cout << "\nRandom value: " << random;**

**// Pivot as middle element k226007**

**count = 0;**

**quickSort(arr, 0, (size - 1) / 2, count);**

**cout << "\n\nSorted array: ";**

**for (int i = 0; i < size; i++)**

**{**

**cout << arr[i] << " ";**

**}**

**cout << "\nPivot middle element count: " << count;**

**// Pivot by median approach**

**count = 0;**

**quickSort(arr, 0, size / 2, count);**

**cout << "\n\nSorted array: ";**

**for (int i = 0; i < size; i++)**

**{**

**cout << arr[i] << " ";**

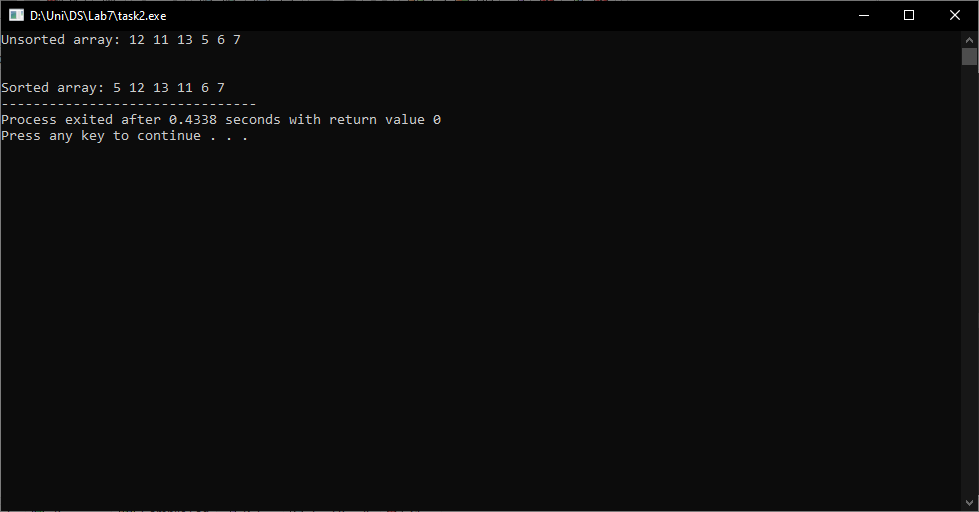
**}**

**cout << "\nPivot by median approach count: " << count;**

**return 0;**

**}**

**Task 2:**



**#include <iostream>**

**using namespace std;**

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

**{**

**int temp;**

**temp = a;**

**a = b;**

**b = temp;**

**}**

**int partition(int arr[], int low, int high)**

**{**

**int mid = low + (high - low) / 2;**

**swap(arr[mid], arr[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[mid]);**

**return (i + 1);**

**}**

**void quickSort(int arr[], int low, int high)**

**{**

**if (low < high)**

**{**

**int pivotIndex = partition(arr, low, high);**

**quickSort(arr, low, pivotIndex - 1);**

**quickSort(arr, pivotIndex + 1, high);**

**}**

**}**

**int main()**

**{**

**int arr[] = {12, 11, 13, 5, 6, 7};**

**int size = sizeof(arr) / sizeof(arr[0]);**

**cout << "Unsorted array: ";**

**for (int i = 0; i < size; i++)**

**{**

**cout << arr[i] << " ";**

**}**

**cout << endl;**

**quickSort(arr, 0, size-1);**

**cout << "\n\nSorted array: ";**

**for (int i = 0; i < size; i++)**

**{**

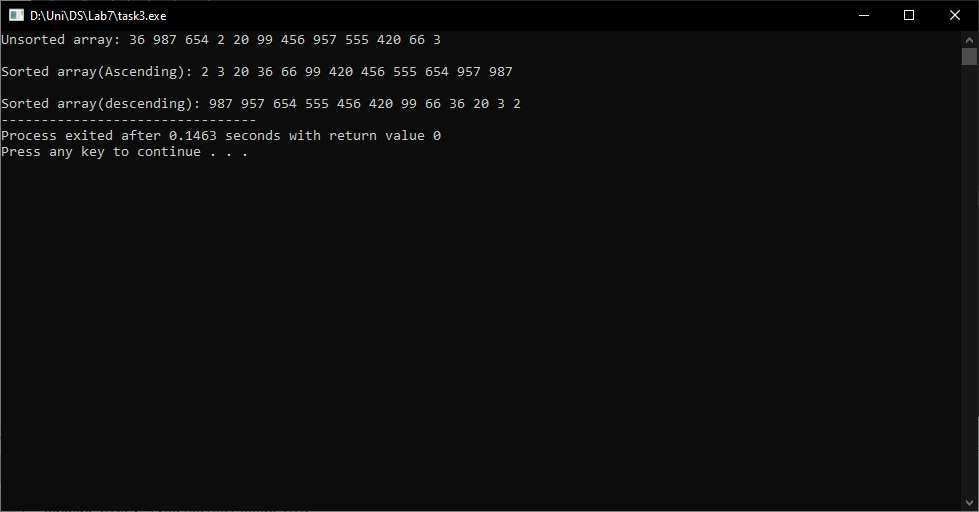
**cout << arr[i] << " ";**

**}**

**return 0;**

**}**

**Task 3:**



**#include <iostream>**

**using namespace std;**

**int get\_max(int arr[], int size)**

**{**

**int max = arr[0];**

**for (int i = 0; i < size; i++)**

**{**

**if (max < arr[i])**

**{**

**max = arr[i];**

**}**

**}**

**return max;**

**}**

**void count\_Sort(int arr[],const int n, int div)**

**{**

**int output[n];**

**int i, count[10] = {0};**

**for ( i = 0; i < n; i++)**

**{**

**count[(arr[i]/ div) % 10]++;**

**}**

**for ( i = 1; i < 10; i++)**

**{**

**count[i] += count[i-1];**

**}**

**for ( i = n-1; i >= 0; i--)**

**{**

**output[count[(arr[i] / div) % 10] - 1] = arr[i];**

**count[(arr[i] / div) % 10]--;**

**}**

**for ( i = 0; i < n; i++)**

**{**

**arr[i] = output[i];**

**}**

**}**

**void radix\_sort(int arr[], int n)**

**{**

**int max = get\_max(arr, n);**

**for (int div = 1; max/div > 0; div\*=10)**

**{**

**count\_Sort(arr, n, div);**

**}**

**}**

**void reverse(int arr[],const int size)**

**{**

**int temp[size];**

**int j = 0;**

**for (int i = size-1; i >= 0; i--)**

**{**

**temp[j] = arr[i];**

**j++;**

**}**

**for (int i = 0; i < size; i++)**

**{**

**arr[i] = temp[i];**

**}**

**}**

**void display(int arr[], int size)**

**{**

**for (int i = 0; i < size; i++)**

**{**

**cout << arr[i] << " ";**

**}**

**}**

**int main()**

**{**

**int arr[] = {36,987,654,2,20,99,456,957,555,420,66,3};**

**int size = sizeof(arr) / sizeof(arr[0]);**

**cout << "Unsorted array: ";**

**display(arr, size);**

**radix\_sort(arr, size);**

**cout << "\n\nSorted array(Ascending): ";**

**display(arr, size);**

**reverse(arr, size);**

**cout << "\n\nSorted array(descending): ";**

**display(arr, size);**

**return 0;**

**}**