23k-0638 Lab 8 Class Submission

Task2:

#include <iostream>

using namespace std;

#define N 10

class Runner{

    public:

        string name;

        int time;

        Runner(string s = "", int n = -1){

            name = s;

            time = n;

        }

        void display(){

            cout << "Name = " << name;

            cout << "   Race-time = " << time << endl;

        }

};

void merge(Runner arr[], int left, int mid, int right){

    int i = left;

    int j = mid + 1;

    int k = left;

    Runner temp[N];

    while(i <= mid && j <= right){

        if(arr[i].time < arr[j].time){

            temp[k] = arr[i];

            i++;

            k++;

        }

        else{

            temp[k] = arr[j];

            j++;

            k++;

        }

    }

    while(i <= mid){

        temp[k] = arr[i];

        i++;

        k++;

    }

    while(j <= right){

        temp[k] = arr[j];

        j++;

        k++;

    }

    for(int count = left; count <= right; count++){

        arr[count] = temp[count];

    }

}

void mergesort(Runner arr[], int l, int r){

    if(l < r){

        int mid = (l + r)/2;

        mergesort(arr, l, mid);

        mergesort(arr, mid + 1, r);

        merge(arr, l, mid, r);

    }

}

int main(){

    Runner runners[N] = {

        Runner("Runner1", 120),

        Runner("Runner2", 150),

        Runner("Uzair", 130),

        Runner("Haseeb", 110),

        Runner("Omar", 100),

        Runner("Shehriyar", 105),

        Runner("Usaid", 180),

        Runner("Tehseen", 170),

        Runner("Hassan", 160),

        Runner("ABC", 165),

    };

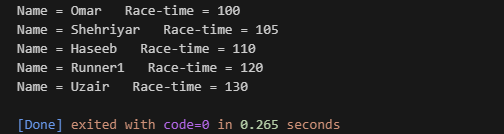
    mergesort(runners, 0, N - 1);

    for(int i = 0; i < 5; i++){

        runners[i].display();

    }

}



Q4)

#include <iostream>

using namespace std;

#define MAX 7

int intArray[MAX] = {10, 1, 2, 3, 4, 5, 6};

void display() {

    cout << "[";

    for (int i = 0; i < MAX; i++) {

        cout << intArray[i] << (i < MAX - 1 ? ", " : "");

    }

    cout << "]\n";

}

void swap(int num1, int num2) {

    int temp = intArray[num1];

    intArray[num1] = intArray[num2];

    intArray[num2] = temp;

}

int partition(int left, int right, int pivotIndex) {

    int pivot = intArray[pivotIndex];

    swap(left, pivotIndex);

    int leftPointer = left + 1;

    int rightPointer = right;

    while (true) {

        while (leftPointer <= right && intArray[leftPointer] < pivot) {

            leftPointer++;

        }

        while (rightPointer > left && intArray[rightPointer] > pivot) {

            rightPointer--;

        }

        if (leftPointer >= rightPointer) {

            break;

        } else {

            cout << "Item swapped: " << intArray[leftPointer] << ", " << intArray[rightPointer] << endl;

            swap(leftPointer, rightPointer);

        }

    }

    cout << "\nPivot swapped: " << intArray[left] << ", " << intArray[rightPointer] << endl;

    swap(left, rightPointer);

    cout << "Updated Array: ";

    display();

    return rightPointer;

}

void quickSort(int left, int right) {

    if (right - left <= 0) {

        return;

    } else {

        int partitionPoint = partition(left, right, left);

        quickSort(left, partitionPoint - 1);

        quickSort(partitionPoint + 1, right);

    }

}

int main() {

    cout << "Input Array: ";

    display();

    quickSort(0, MAX - 1);

    cout << "\nOutput Array: ";

    display();

}

