**Muhammad Ahmed Fraz**

**332779**

**BSCS-10C**

**LAB7 DSA**

**Task1:**

**Code:**

#include<iostream>

#include<conio.h>

#include<time.h>

#include<stdlib.h>

using namespace std;

int main(){

int numberOfTimes=0;

cout << "Enter for how many times you want to roll a dice : " << endl;

cin >> numberOfTimes;

srand(time(NULL));

for (int counter = 0; counter < numberOfTimes; counter++){

for (int i = 0; i < 6; i++)

{

cout << "Dice: "<<rand() % 6 + 1 << endl;

}

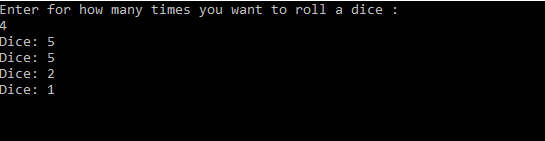
}

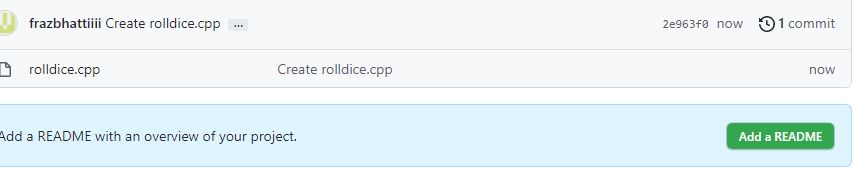
getchar();

getchar();

}

**Output:**





**Task 2:**

#include<iostream>

#include<conio.h>

#include<time.h>

#include<stdlib.h>

#include<vector>

#include<algorithm>

using namespace std;

void swap(int\* ptr1, int\* ptr2)

{

int temp = \*ptr1;

\*ptr1 = \*ptr2;

\*ptr2 = temp;

}

// A function to implement bubble sort

bool movMin(vector<int>& in, vector<int>& out)

{

int i, j;

for (i = 0; i < in.size() - 1; i++)

// Last i elements are already in place

for (j = 0; j < in.size() - i - 1; j++)

if (in[j] > in[j + 1])

swap(&in[j], &in[j + 1]);

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

out.push\_back(in.at(i));

}

cout << endl;

cout << "The sorted array after pushing the last element is : " << endl;

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

cout << out[i] << " ";

}

getchar();

return true;

}

bool testMovMIn() {

srand(time(NULL));

vector<int> test;

vector<int> copyTest;

for (int i = 0; i < 50; i++) {

//Pushing 100 random elements in the array

test.push\_back(rand() % 100 + 1);

}

cout << "the array before sorting is : " << endl;

for (int i = 0; i < 50; i++) {

//Pushing 100 random elements in the array

cout << test[i] << " ";

}

cout << endl;

cout << "The sorted array is : " << endl;

sort(test.begin(), test.end());//sorting the array by stl sort function

for (int i = 0; i < 50; i++) {

//Pushing 100 random elements in the array

cout << test[i] << " ";

}

test.push\_back(rand() % 100 + 1);//pushing another random element in the array

cout << endl;

cout << "The array with pushing the last element is : " << endl;

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

//Pushing 100 random elements in the array

cout << (test[i]) << " ";

}

//Copying into the copytest and also running the movMin function

movMin(test, copyTest);//Calling the function to check the test cases

return true;

}

int main() {

vector<int> arrayInput;

vector<int> dummy;

int input, size;

cout << "Enter the size of the array: " << endl;

cin >> size;

cout << "Enter the input elements to sort: " << endl;

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

cin >> input;

arrayInput.push\_back(input);

}

cout << endl;

cout << "The array which you enter is : " << endl;

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

cout << arrayInput[i]<<" ";

}

cout << endl;

movMin(arrayInput, dummy);

testMovMIn();

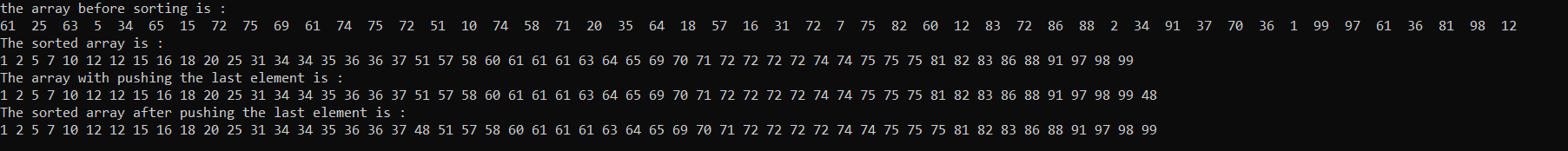
getchar();

}

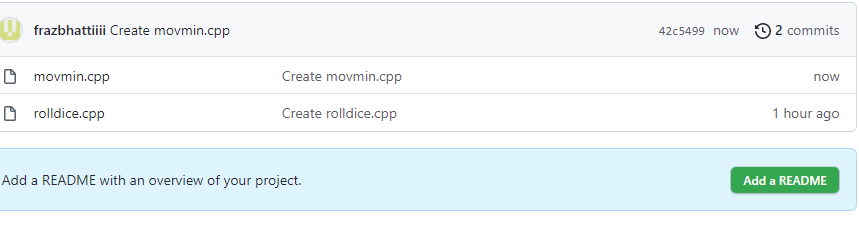
**Output:**

Text

Description automatically generated

****

**GitHub Commit History:**



**Task 3:**

**Code:**

#include<iostream>

#include<conio.h>

#include<time.h>

#include<stdlib.h>

#include<vector>

#include<algorithm>

#include <chrono>

using namespace std;

using namespace std::chrono;

void swap(int \*ptr1, int \*ptr2)

{

int temp = \*ptr1;

\*ptr1 = \*ptr2;

\*ptr2 = temp;

}

// A function to implement bubble sort

bool movMin(vector<int>&in, vector<int>&out)

{

for (int i = 0; i < in.size() - 2; i++)

{

if (in[i] > in[i+ 1])

{

int temp = in[i];

in[i] = in[i + 1];

in[i + 1] = temp;

i = -1;

}

}

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

out.push\_back(in.at(i));

}

cout << "The sorted array without changing the last element" << endl;

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

cout << out[i] << " ";

}

return true;

}

bool testMovMIn(int n){

srand(time(NULL));

vector<int> test;

vector<int> copyTest;

for (int i = 0; i <n; i++){

//Pushing 100 random elements in the array

test.push\_back(rand() % 100 + 1);

}

cout << "the array before sorting is : " << endl;

for (int i = 0; i < n; i++){

//Pushing 100 random elements in the array

cout << test[i]<<" ";

}

cout << endl;

cout << "The sorted array is : " << endl;

sort(test.begin(), test.end());

for (int i = 0; i < n; i++){

//Pushing 100 random elements in the array

cout << test[i]<<" ";

}

cout << endl;

test.push\_back(rand() % 100 + 1);

//Copying into the copytest and also running the movMin function

movMin(test, copyTest);

return true;

}

bool ChecktestMovMIn(vector<int> test){

int n = test.size();

vector<int> copyTest;

cout << "The array before all operation is: " << endl;

for (int i = 0; i < n; i++){

//Pushing 100 random elements in the array

cout << test[i] << " ";

}

cout << endl;

test.push\_back(rand() % 100 + 1);

//Copying into the copytest and also running the movMin function

movMin(test, copyTest);

return true;

}

bool bestCase(int n){

srand(time(NULL));

vector<int> test;

vector<int> copyTest;

for (int i = 0; i <n; i++){

//Pushing 100 random elements in the array

test.push\_back(rand() % 100 + 1);

}

//For this case the array will be sorted first and then we will check the case

sort(test.begin(), test.end());

ChecktestMovMIn(test);

return true;

}

bool worstCase(int n){

srand(time(NULL));

vector<int> test;

vector<int> copyTest;

for (int i = 0; i <n; i++){

//Pushing 100 random elements in the array

test.push\_back(rand() % 100 + 1);

}

for (int i = 0; i < test.size() - 2; i++)

{

if (test[i] < test[i + 1])

{

int temp = test[i];

test[i] = test[i + 1];

test[i + 1] = temp;

i = -1;

}

}

//For this case the array will be in descending order

ChecktestMovMIn(test);

return true;

}

int main(){

vector<int> arrayInput;

vector<int> dummy;

int input,size;

cout << "Enter the size of the array: " << endl;

cin >> size;

cout << "Enter the input elements to sort: " << endl;

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

cin >> input;

arrayInput.push\_back(input);

}

cout << endl;

cout << "The array which you enter is : " << endl;

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

cout<< arrayInput[i]<<" ";

}

cout << endl;

movMin(arrayInput, dummy);

testMovMIn(10000);

Test Cases to check different values

10 ,100 ,1000, 10000

auto start = high\_resolution\_clock::now();

testMovMIn(100);

auto stop = high\_resolution\_clock::now();

auto duration = duration\_cast<microseconds>(stop - start);

//From geeks for geeks this algo function

cout << endl;

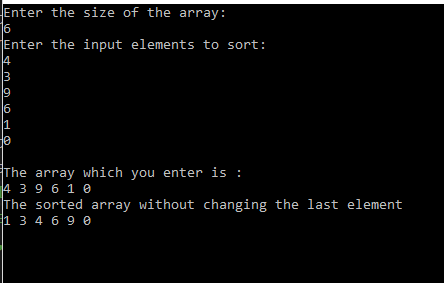
cout << endl;

cout << duration.count() <<" microseconds"<< endl;

getchar();

getchar();

}



**Test Cases:**

**10 inputs:**

**Text

Description automatically generated**

**100 inputs:**

**A screen shot of a computer

Description automatically generated with low confidence**

**1000 inputs:**

**A picture containing background pattern

Description automatically generated**

**Time:**

**For 50 inputs it will take 19369 microseconds.**

**A picture containing graphical user interface

Description automatically generated**

**Best Case:  
In which the array is sorted first:**

**9040220 microseconds**

**Text

Description automatically generated**

**Worst Case:**

**It takes a lot of time to sort even in the 100 elements**

**13925834 microseconds**

**For worst case** 100000 it takes a lot time that even for minutes there is no output on console

**A picture containing text

Description automatically generated**

**Average Case:**

**5897759 is the average time for 1000 inputs which is still very greater and at 100000 it also takes a lot of time.**

**A screen shot of a computer

Description automatically generated with low confidence**

**Github commit:**

