**Name: ABDUL GHAFFAR KALHORO**

**Registration: 194699**

**Class: BSCS-6C**

**Date: 15-Dec-17**

**LAB# 10 DATA STRUCTURES AND ALGORITHM**

**implementing insertion sort and finding worst case complexity for n = 100000**

**TASK# (a)**

**Finding time complexity of insertion sort for unsorted random numbers**

**Source code:**

//task3- (a) worst case time for random numbers sorting

#include <iostream>

#include <fstream>

#include <string>

#include <ctime>

using namespace std;

#define size = 100000;

void insertionSortA(int array\_words[], int sizes) {

int key;

int a;

for (int b = 1; b<sizes; b++) {

key = array\_words[b];

for (a = b - 1; a >= 0 & array\_words[a]>key; a--)

array\_words[a + 1] = array\_words[a];

array\_words[a + 1] = key;

}

}

int main(){

int find\_avg[10];

srand(time(NULL));

int myArray[100000];

int j = 0;

cout << "unsorted numbers: ";

while (j < 10){

for (int i = 0; i < 100000; i++){

myArray[i] = (rand() % 100000) + 1;

}

clock\_t begin = clock();

insertionSortA(myArray, 100000);

clock\_t end = clock();

double elapsed\_seconds = double(end - begin) / CLOCKS\_PER\_SEC \* 1000;

cout << endl << "random numbers sorted in : " << elapsed\_seconds << " millisecs."

<< endl;

find\_avg[j] = elapsed\_seconds;

j++;

}//end while

double sum = 0;

int a = 0;

while (a < 10) {

sum = sum + find\_avg[a];

a++;

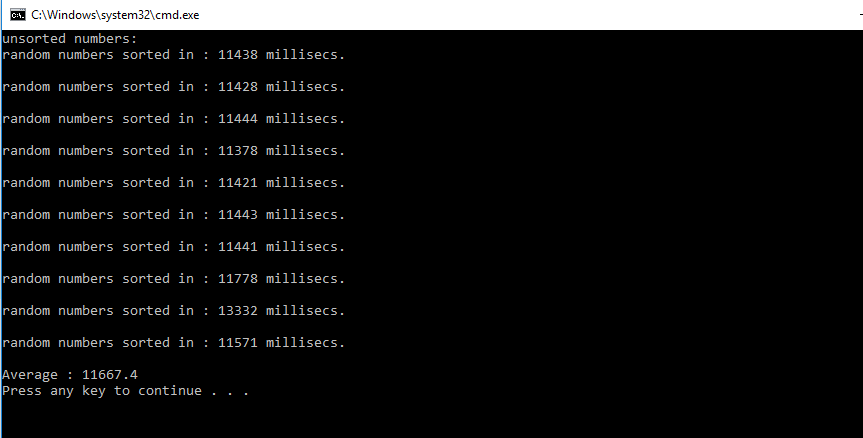
}

cout << endl << "Average : " << sum / 10 << endl;

return 0;

}

**OUTPUT**

****

**TASK# (b)**

**Finding time complexity of insertion sort for ascending ordered random numbers**

**Source code:**

//task3- (b) worst case time for ascending numbers sorting

#include <iostream>

#include <fstream>

#include <string>

#include <ctime>

using namespace std;

#define size = 100000;

void insertionSortA(int array\_words[], int sizes) {

int key;

int g;

for (int h = 1; h<sizes; h++) {

key = array\_words[h];

for (g = h - 1; g >= 0 & array\_words[g]>key; g--)

array\_words[g + 1] = array\_words[g];

array\_words[g + 1] = key;

}

}

int main(){

int my\_numbers[10];

int y = 99999;

srand(time(NULL));

int array[100000];

int array2[100000];

int j = 0;

cout << "sorted numbers in ascending order: ";

while (j < 10){

for (int i = 0; i < 100000; i++){

array[i] = (rand() % 100000) + 1;

}

//before calculating the time, first convert the array of random numbers in ascending sort

insertionSortA(array, 100000);

clock\_t begin = clock();

insertionSortA(array2, 100000);

clock\_t end = clock();

double elapsed\_seconds = double(end - begin) / CLOCKS\_PER\_SEC \* 1000;

cout << endl << "sorted numbers sorted in : " << elapsed\_seconds << " millisecs."

<< endl;

my\_numbers[j] = elapsed\_seconds;

j++;

}//end while

double sum = 0;

int a = 0;

while (a < 10) {

sum = sum + my\_numbers[a];

a++;

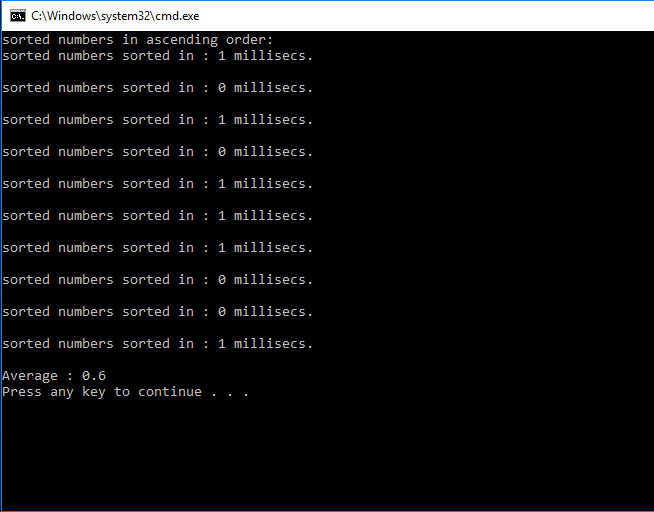
}

cout << endl << "Average : " << sum / 10 << endl;

return 0;

}

**OUTPUT**

****

**TASK# (c)**

**Finding time complexity of insertion sort for descending ordered random numbers**

**Source code:**

//task3(c)- worst case time for random numbers sorted in descending order.

#include <iostream>

#include <fstream>

#include <string>

#include <ctime>

using namespace std;

//insertion sort algorithm

void insertionSort(int array[], int sizes) {

int key;

int a;

for (int b = 1; b<sizes; b++) {

key = array[b];

for (a = b - 1; a >= 0 & array[a]>key; a--)

array[a + 1] = array[a];

array[a + 1] = key;

}

}

int main(){

int findAver[10];

int y = 99999;

srand(time(NULL));

int array[100000];

int j = 0;

int num;

cout << "sorted numbers in descending order: ";

while (j < 10){

num = (rand() % 10) + 1;

for (int i = 99999; i >= 0; i--){

array[i] = num;

num++;

}

clock\_t begin = clock();

insertionSort(array, 100000);

clock\_t end = clock();

double elp\_sec = double(end - begin) / CLOCKS\_PER\_SEC \* 1000;

cout << endl << j + 1 << ") sorting time : " << elp\_sec << " millisecs."

<< endl;

findAver[j] = elp\_sec;

j++;

}//end while

double sum = 0;

int a = 0;

while (a < 10) {

sum = sum + findAver[a];

a++;

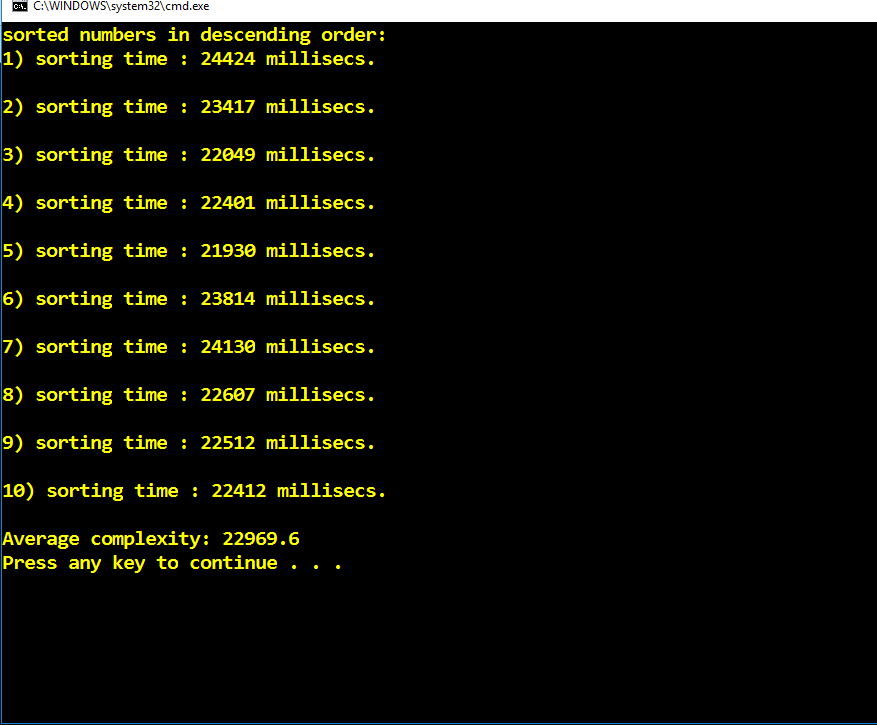
}

cout << endl << "Average complexity: " << sum / 10 << endl;

return 0;

}

**OUTPUT:**

****

**Answer to the Questions**

From the result of above programs, we have concluded that:

1. The best case of the sorting for insertion sort is that when all the data is already sorted as founded in the sorted ascending random numbers
2. The average case is founded in the sorting of random numbers which is when we take total numbers = 100000 in array then the time complexity of the insertion sort becomes 11.66 seconds
3. Now for the worst case there is the pattern in which the largest element is stored in the start of array and we need to move it to the end as done in the descendent sorting random numbers in which total numbers = 100000 took 22.9 seconds.