Name	RITIK SINGH
UID no.	2021700061
Experiment No.	1b

AIM:	To finding the running time of an algorithm.
Program 1	
PROBLEM STATEMENT :	Implement two sorting algorithms namely Insertion and Selection sort methods. Compare these algorithms based on time and space complexity. Time required to sorting algorithms can be performed using high_resolution_clock::now() under namespace std::chrono. You have togenerate1,00,000 integer numbers using C/C++ Rand function and save them in a text file. Boththe sorting algorithms uses these 1,00,000 integer numbers as input as follows. Each sorting algorithm sorts a block of 100 integers numbers with array indexes numbers A[0..99], A[0..199], A[0..299],..., A[0..99999]. You need to use high_resolution_clock::now() function to find the time required for 100, 200, 300.... 100000 integer numbers. Finally, compare two algorithms namely Insertion and Selection by plotting the time required to sort 100000 integers using LibreOffice Calc/MS Excel. The x-axis of 2-D plot represents the block no. of 1000 blocks. Th y-axis of 2-D plot represents the tunning time to sort 1000 blocks of 100,200,300,...,100000 integer numbers.
PROGRAM:	see below

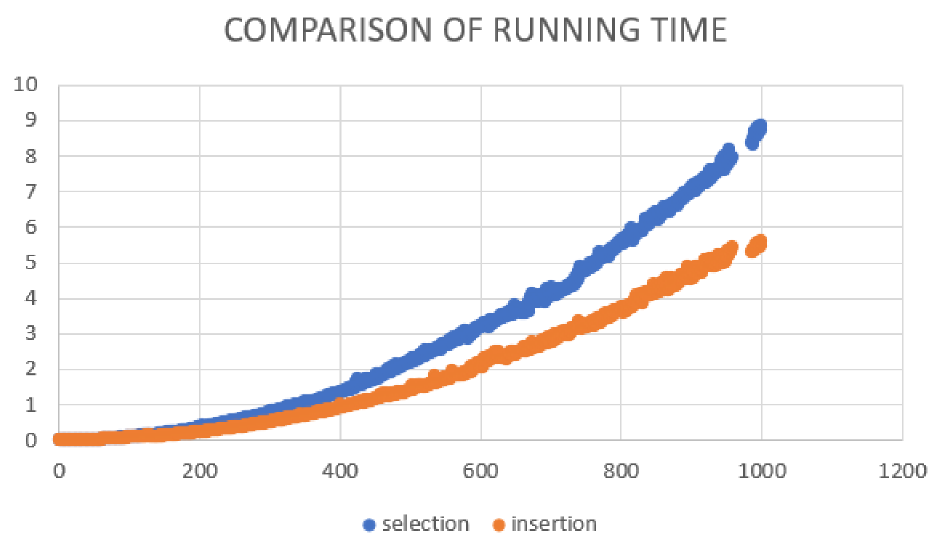
```c
#include <stdio.h>
#include <math.h>
#include <conio.h>
#include <stdlib.h>
#include <time.h>

void getInput()
{
  FILE *fp;
  fp = fopen("input.text","w");
  for(int i=0;i<100000;i++)
  fprintf(fp,"%d ",rand()%100000);
  fclose(fp);
}
```

```
void insertionSort(int arr[], int size) {
    for (int i = 1; i < size; i++) {
        int key = arr[i];
        int j = i - 1;
        while (key < arr[j] && j >= 0) {
            arr[j + 1] = arr[j];
            --j;
        }
        arr[j + 1] = key;
    }
}

void selectionSort(int arr[], int len){
    int minIndex, temp;
    for(int i=0; i<len; i++){
        minIndex = i;
        for(int j=i+1; j<len; j++){
            if(arr[j] < arr[minIndex]){
                minIndex = j;
            }
        }
        temp = arr[minIndex];
        arr[minIndex] = arr[i];
        arr[i] = temp;
    }
}

int main(){

    getInput();
    FILE *rt, *tks;
    int a=99;
    int arrNums[100000];
    clock_t t;
    rt = fopen("input.text", "r");
    tks = fopen("iTimes.txt", "w");
    for(int i=0; i<300; i++){
        for(int j=0; j<=a; j++){
            fscanf(rt, "%d", &arrNums[j]);
        }
        t = clock();
        insertionSort(arrNums, a+1);
        t = clock() - t;
        double time_taken = ((double)t)/CLOCKS_PER_SEC;
```

```c
        fprintf(tks, "time taken for %d iteration is %Lf\n", (i+1), time_taken)
        printf("%d\t%lf\n", (i+1), time_taken);
        a = a + 100;
        fseek(rt, 0, SEEK_SET);
    }
    fclose(tks);
    tks = fopen("STimes.txt", "w");
    a=99;
    for(int i=0; i<300; i++){
        for(int j=0; j<=a; j++){
            fscanf(rt, "%d", &arrNums[j]);
        }
        t = clock();
        selectionSort(arrNums, a+1);
        t = clock() - t;
        double time_taken = ((double)t)/CLOCKS_PER_SEC;
        fprintf(tks, "time taken for %d iteration is %Lf\n", (i+1), time_taken)
        printf("%d\t%lf\n", (i+1), time_taken);
        a = a + 100;
        fseek(rt, 0, SEEK_SET);
    }
    fclose(tks);
    fclose(rt);
    return 0;
}
```

RESULT:

GRAPH :

	
CONCLUSION:	I HAVE LEARNT AND IMPLEMENT THE INSERTION AND SELECTION SORT ALGORITHM AND COMPARE THE TIME COMPLEXITY BETWEEN THESE TWO ALGORITHM. INSERTION SORT HAS BETTER TIME COMPLEXITY THAN SELECTION