NAME	RITIK SINGH
UID	2021700061
CLASS	SY CSE(DS)
BATCH	D

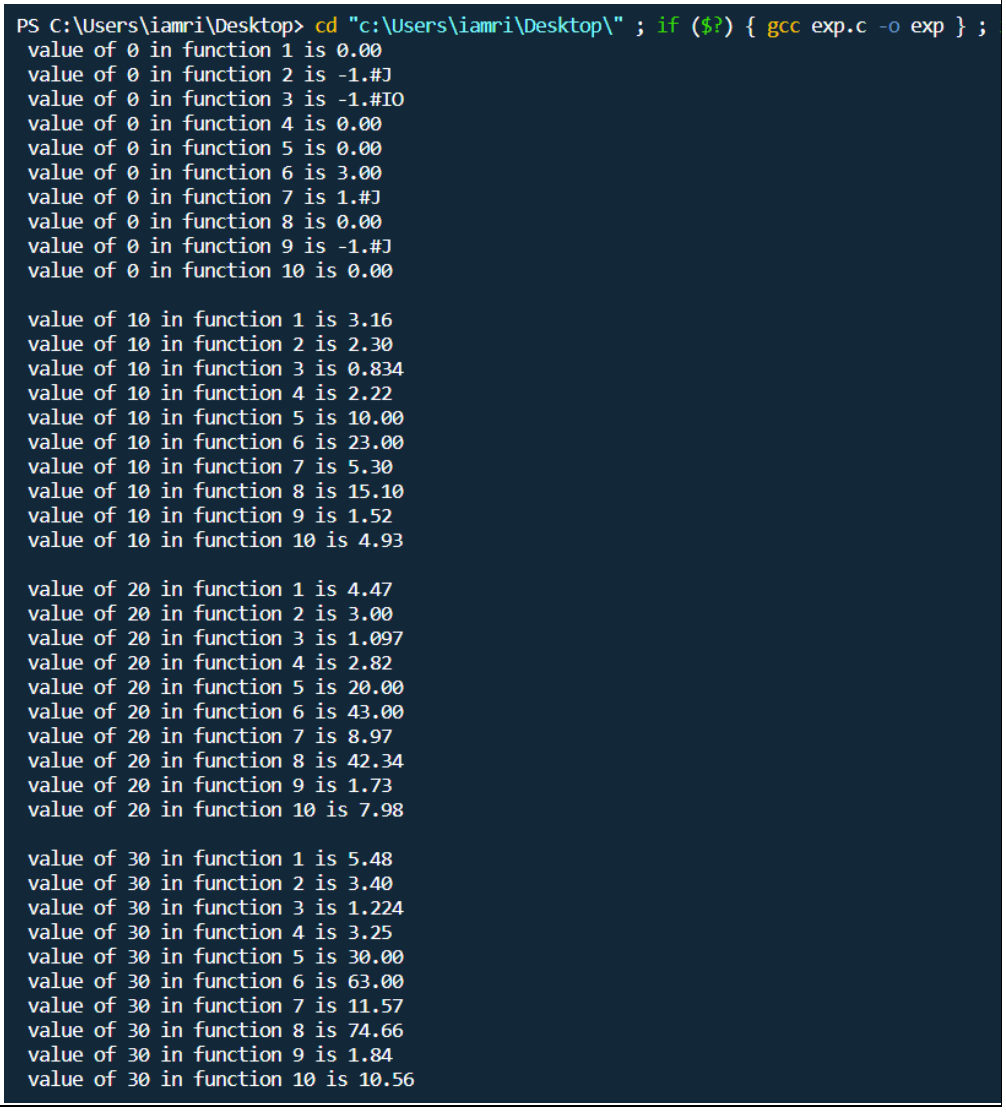
Exp1-A

AIM	To implement the various functions e.g. linear, non-linear, quadratic, exponential etc.

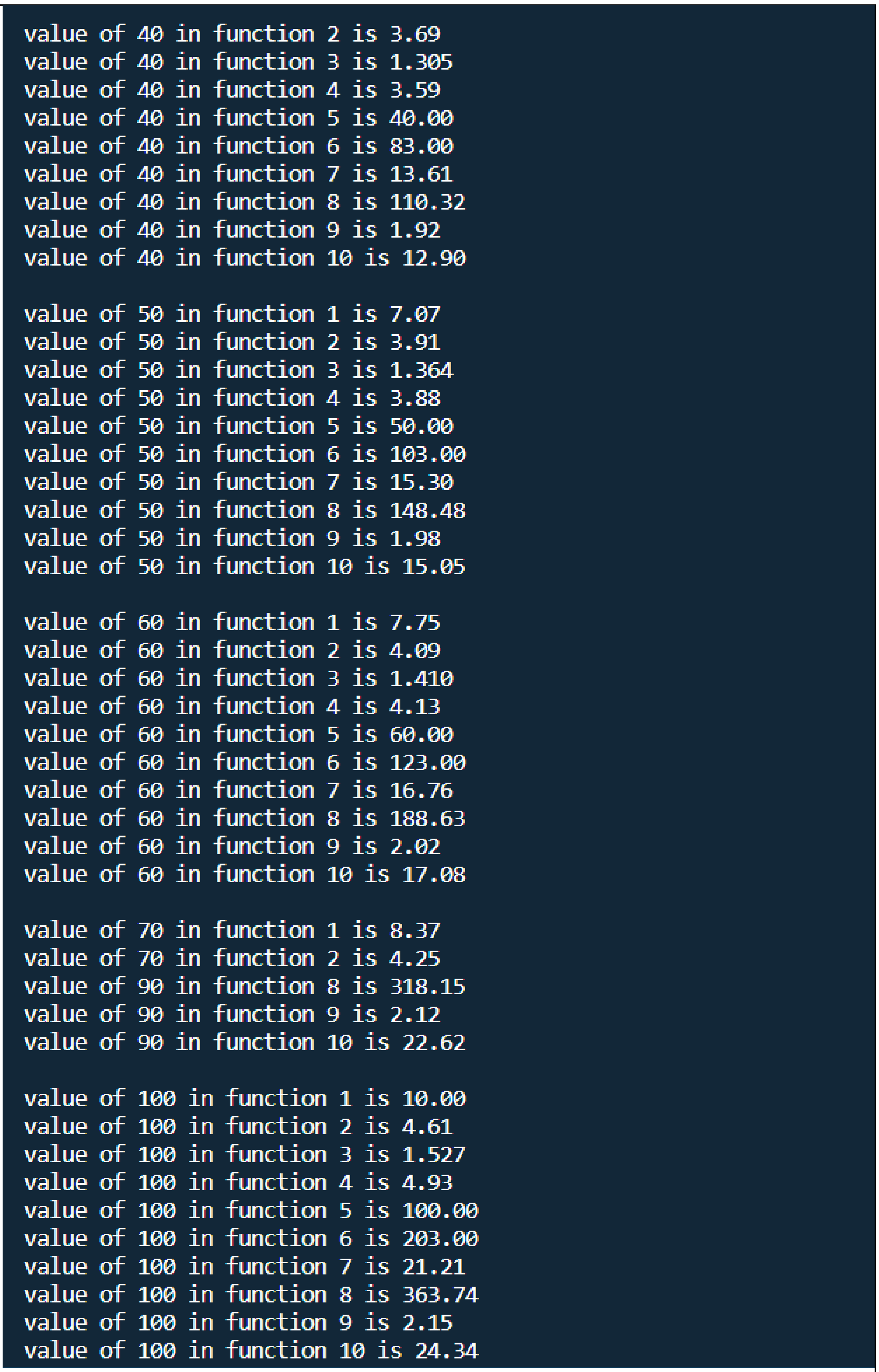
PROGRAM

```c
#include<stdio.h>
#include<math.h>

  double fac(int n)
  {
  if (n==0)
   return 1;
   return n*fac(n-1);
  }

  float fun1(int a)
  {
    return sqrt(a);
  }
  float fun2(int a)
  {
    return log(a);
  }
  float fun3(int a)
  {
    return log(log(a));
  }
  float fun4(int a)
  {
    return pow(sqrt(2),log(a));
  }
  float fun5(int a)
  {
    return a;
  }
  float fun6(int a)
  {
    return 2*a+3;;
  }
  float fun7(int a)
  {
    return pow(log(a),2);
  }
  float fun8(int a)
  {
    return log(fac(a));
  }
  float fun9(int a)
  {
    return sqrt(log(a));
  }
  float fun10(int a)
  {
    return pow(2,log(a));
  }


 int main()
 {



   for ( int i=0;i<=100;i=i+10)
    {

     printf(" value of %d in function 1 is %0.2f\n",i,
fun1(i)); //
     printf(" value of %d in function 2 is %0.2f\n",i,
fun2(i));
     printf(" value of %d in function 3 is %0.3f\n",i,
fun3(i));
     printf(" value of %d in function 4 is %0.2f\n",i,
fun4(i));
     printf(" value of %d in function 5 is %0.2f\n",i,
fun5(i));
     printf(" value of %d in function 6 is %0.2f\n",i,
fun6(i));
     printf(" value of %d in function 7 is %0.2f\n",i,
fun7(i));
     printf(" value of %d in function 8 is %0.2f\n",i,
fun8(i));
     printf(" value of %d in function 9 is %0.2f\n",i,
fun9(i));
     printf(" value of %d in function 10 is %0.2f\n",i,
fun10(i));
     printf(" value of %d factorial is %0.2f\n",i, fac(i));
     printf("\n");

    }

 }
```

OUTPUT	

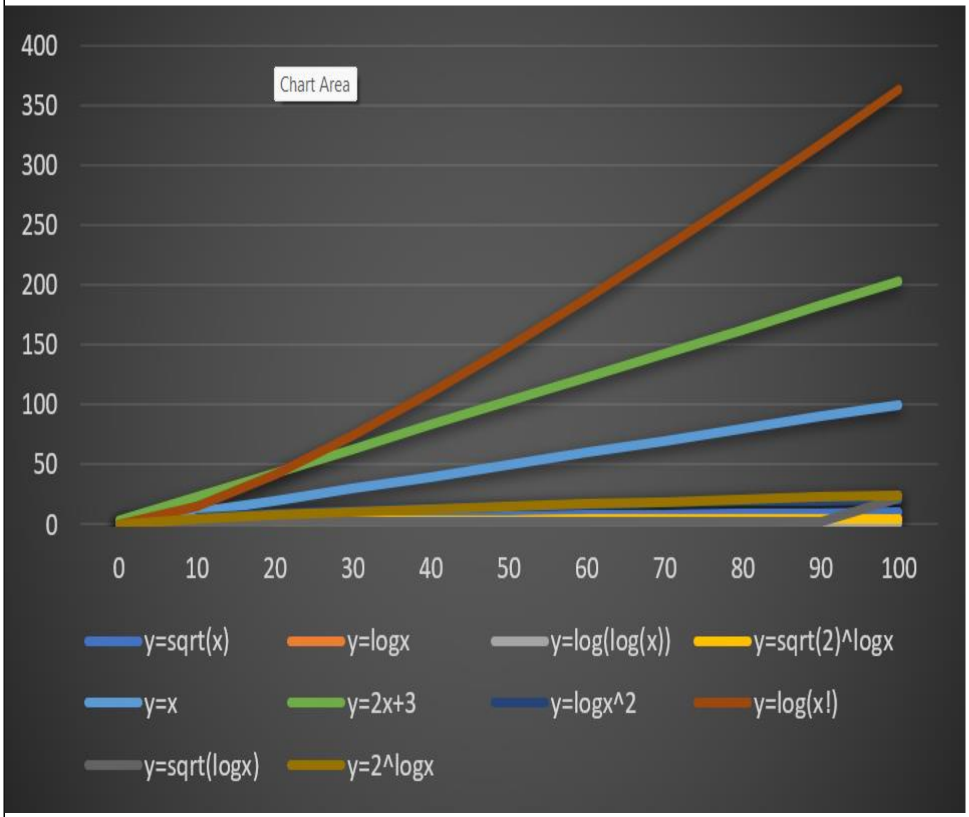
```
PS C:\Users\iamri\Desktop> cd "c:\Users\iamri\Desktop\" ; if ($?) { gcc exp.c -o exp } ;
 value of 0 in function 1 is 0.00
 value of 0 in function 2 is -1.#J
 value of 0 in function 3 is -1.#IO
 value of 0 in function 4 is 0.00
 value of 0 in function 5 is 0.00
 value of 0 in function 6 is 3.00
 value of 0 in function 7 is 1.#J
 value of 0 in function 8 is 0.00
 value of 0 in function 9 is -1.#J
 value of 0 in function 10 is 0.00

 value of 10 in function 1 is 3.16
 value of 10 in function 2 is 2.30
 value of 10 in function 3 is 0.834
 value of 10 in function 4 is 2.22
 value of 10 in function 5 is 10.00
 value of 10 in function 6 is 23.00
 value of 10 in function 7 is 5.30
 value of 10 in function 8 is 15.10
 value of 10 in function 9 is 1.52
 value of 10 in function 10 is 4.93

 value of 20 in function 1 is 4.47
 value of 20 in function 2 is 3.00
 value of 20 in function 3 is 1.097
 value of 20 in function 4 is 2.82
 value of 20 in function 5 is 20.00
 value of 20 in function 6 is 43.00
 value of 20 in function 7 is 8.97
 value of 20 in function 8 is 42.34
 value of 20 in function 9 is 1.73
 value of 20 in function 10 is 7.98

 value of 30 in function 1 is 5.48
 value of 30 in function 2 is 3.40
 value of 30 in function 3 is 1.224
 value of 30 in function 4 is 3.25
 value of 30 in function 5 is 30.00
 value of 30 in function 6 is 63.00
 value of 30 in function 7 is 11.57
 value of 30 in function 8 is 74.66
 value of 30 in function 9 is 1.84
 value of 30 in function 10 is 10.56
```



```
value of 40 in function 2 is 3.69
value of 40 in function 3 is 1.305
value of 40 in function 4 is 3.59
value of 40 in function 5 is 40.00
value of 40 in function 6 is 83.00
value of 40 in function 7 is 13.61
value of 40 in function 8 is 110.32
value of 40 in function 9 is 1.92
value of 40 in function 10 is 12.90

value of 50 in function 1 is 7.07
value of 50 in function 2 is 3.91
value of 50 in function 3 is 1.364
value of 50 in function 4 is 3.88
value of 50 in function 5 is 50.00
value of 50 in function 6 is 103.00
value of 50 in function 7 is 15.30
value of 50 in function 8 is 148.48
value of 50 in function 9 is 1.98
value of 50 in function 10 is 15.05

value of 60 in function 1 is 7.75
value of 60 in function 2 is 4.09
value of 60 in function 3 is 1.410
value of 60 in function 4 is 4.13
value of 60 in function 5 is 60.00
value of 60 in function 6 is 123.00
value of 60 in function 7 is 16.76
value of 60 in function 8 is 188.63
value of 60 in function 9 is 2.02
value of 60 in function 10 is 17.08

value of 70 in function 1 is 8.37
value of 70 in function 2 is 4.25
value of 90 in function 8 is 318.15
value of 90 in function 9 is 2.12
value of 90 in function 10 is 22.62

value of 100 in function 1 is 10.00
value of 100 in function 2 is 4.61
value of 100 in function 3 is 1.527
value of 100 in function 4 is 4.93
value of 100 in function 5 is 100.00
value of 100 in function 6 is 203.00
value of 100 in function 7 is 21.21
value of 100 in function 8 is 363.74
value of 100 in function 9 is 2.15
value of 100 in function 10 is 24.34
```

GRAPHS

values	y=sqrt(x)	y=logx	y=log(log(	y=sqrt(2)^	y=x	y=2x+3	y=logx^2	y=log(x!)	y=sqrt(log	y=2^logx
0	0		0	0	0	3		0	0	0
10	3.16	2.3	0.834	2.22	10	23	5.3	15.1	1.52	4.93
20	4.47	3	1.097	2.82	20	43	8.97	42.34	1.73	7.98
30	5.48	3.4	1.224	3.25	30	63	11.57	74.66	1.84	10.56
40	6.32	3.69	1.305	3.59	40	83	13.61	110.32	1.92	12.9
50	7.07	3.91	1.364	3.88	50	103	15.3	148.48	1.98	15.05
60	7.75	4.09	1.41	4.13	60	123	16.76	188.63	2.02	17.08
70	8.37	4.25	1.447	4.36	70	143	18.05	230.44	2.06	19.01
80	8.94	4.38	1.478	4.57	80	163	19.2	273.667	2.09	20.85
90	9.49	4.5	1.504	4.76	90	183	20.25	318.15	2.12	22.62
100	10	4.61	1.527	4.93	100	203	21.21	363.74	21.5	24.34



CONCLUSION	1)Among all,2^2^x+1 increases to infinity the fastest. 2)amongst all,log(log(x)) tends to zero as the number increases from 0-100 3)x.logx function does not saturate as the other log functions. 4)logx and x^log(logx) intersects 5)Other Increasing and decreasing nature of the graphs is observed.

Exp1-B

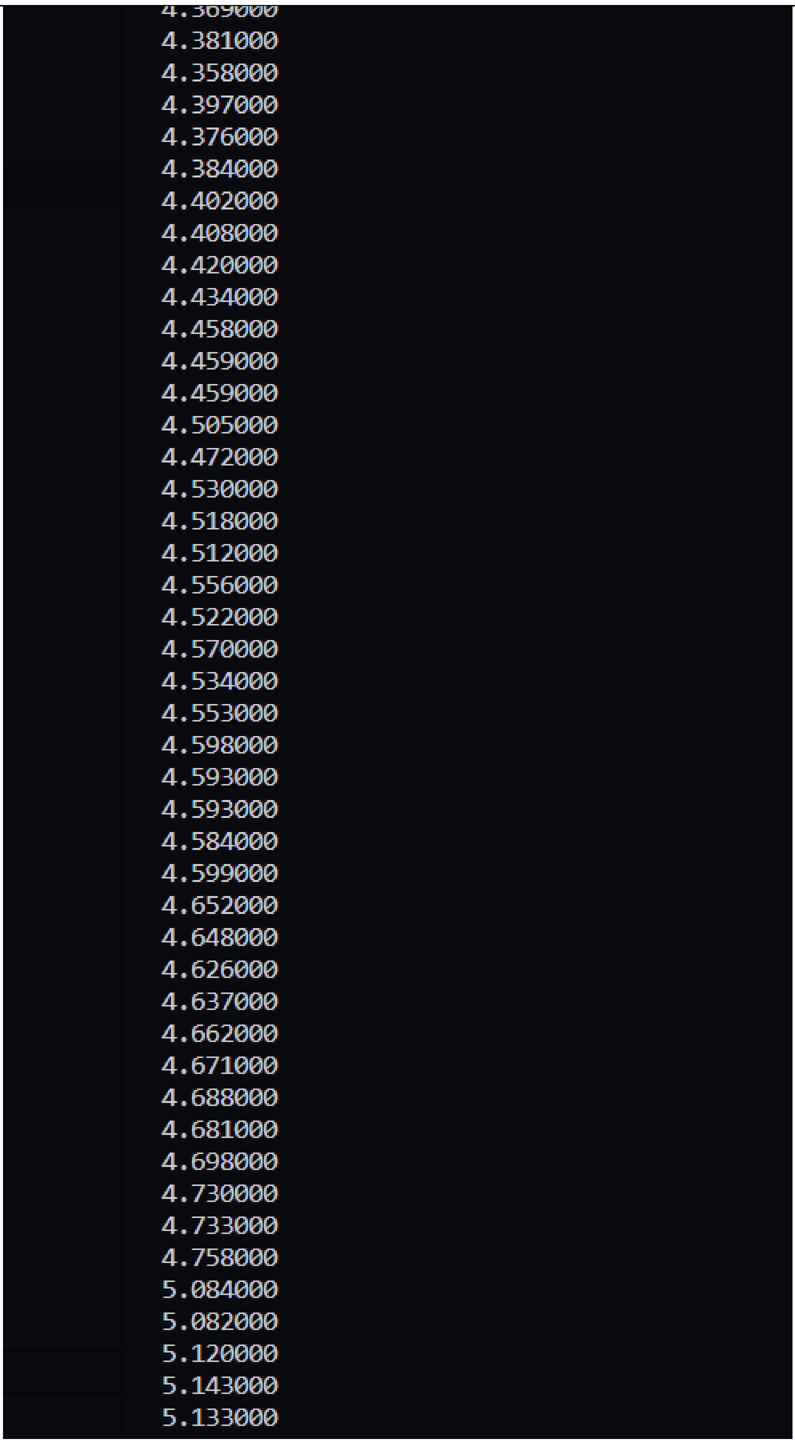
AIM	Experiment on finding the running time of an algorithm.(Insertion and Selection sort)
ALGORITHM	**Insertion sort-** **Step 1 -** If the element is the first element, assume that it is already sorted. Return 1. **Step2 -** Pick the next element, and store it separately in a **key.** **Step3 -** Now, compare the **key** with all elements in the sorted array. **Step 4 -** If the element in the sorted array is smaller than the current element, then move to the next element. Else, shift greater elements in the array towards the right. **Step 5 -** Insert the value. **Step 6 -** Repeat until the array is sorted. **Selection sort-** **Step 1** – Set MIN to location 0 **Step 2** – Search the minimum element in the list **Step 3** – Swap with value at location MIN **Step 4** – Increment MIN to point to next element **Step 5** – Repeat until list is sorted

PROGRAM

```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);
}
```

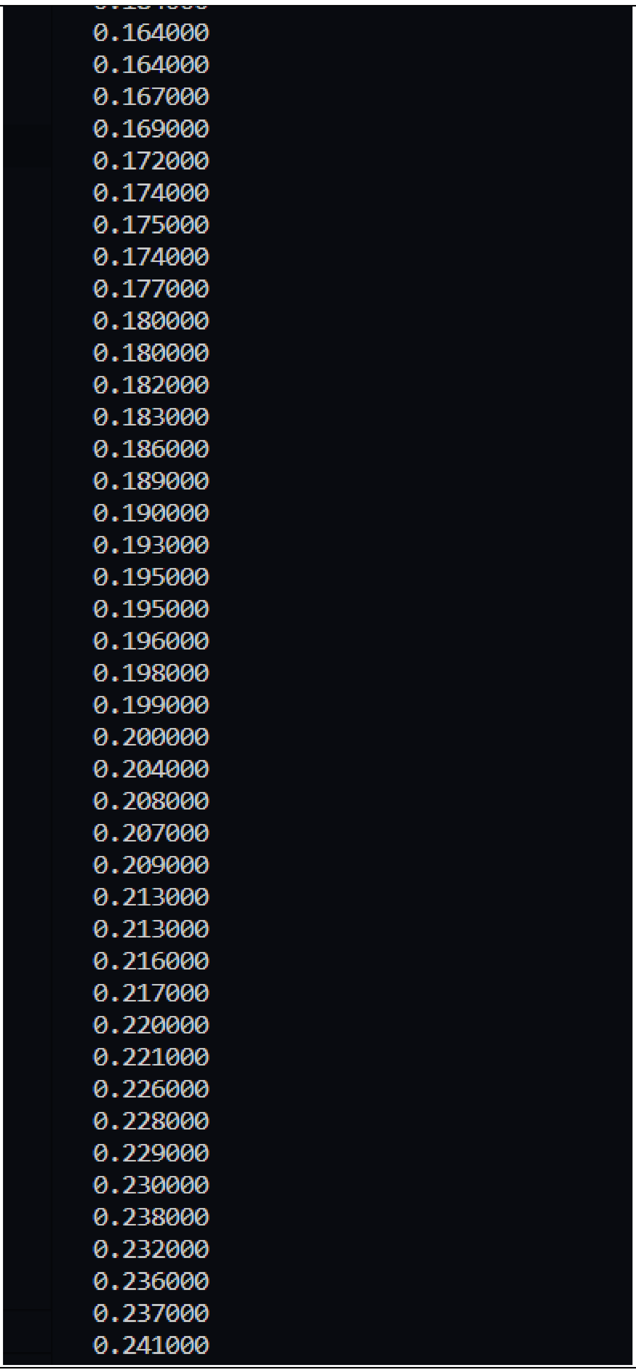
```c
void insertionSort(int arr[], int size) {
    for (int i = 1; i < size; i++) {
        int key = arr[i];
        int j = i - 1;
        while (key < arr[j] && j >= 0) {
            arr[j + 1] = arr[j];
            --j;
        }
        arr[j + 1] = key;
    }
}

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;
        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	**Selection Sort-**

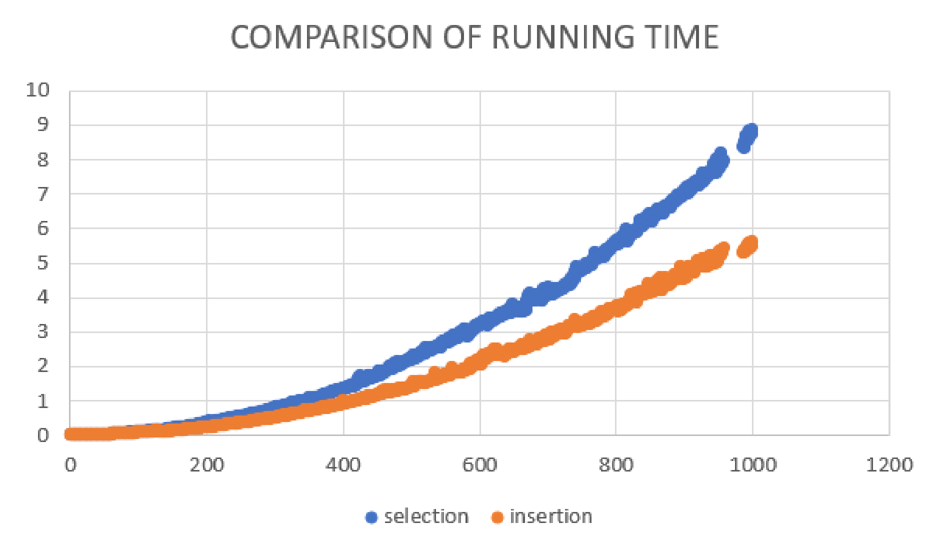


```
4.369000
4.381000
4.358000
4.397000
4.376000
4.384000
4.402000
4.408000
4.420000
4.434000
4.458000
4.459000
4.459000
4.505000
4.472000
4.530000
4.518000
4.512000
4.556000
4.522000
4.570000
4.534000
4.553000
4.598000
4.593000
4.593000
4.584000
4.599000
4.652000
4.648000
4.626000
4.637000
4.662000
4.671000
4.688000
4.681000
4.698000
4.730000
4.733000
4.758000
5.084000
5.082000
5.120000
5.143000
5.133000
```

Insertion sort-

```
0.164000
0.164000
0.167000
0.169000
0.172000
0.174000
0.175000
0.174000
0.177000
0.180000
0.180000
0.182000
0.183000
0.186000
0.189000
0.190000
0.193000
0.195000
0.195000
0.196000
0.198000
0.199000
0.200000
0.204000
0.208000
0.207000
0.209000
0.213000
0.213000
0.216000
0.217000
0.220000
0.221000
0.226000
0.228000
0.229000
0.230000
0.238000
0.232000
0.236000
0.237000
0.241000
```

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