NAME:	SANIYA BANGARE
UID:	2021300009
SUBJECT:	DAA
EXPERIMENT NO:	1
DATE OF SUBMISSION:	09/02/23
AIM:	**1.A:To implement the various functions e.g. linear, non-linear, quadratic, exponential etc.** **1-B: Experiment on finding the running time of an algorithm.**
Theory: **ALGORITHM:**	The understanding of running time of algorithms is explored by implementing two basic sorting algorithms namely Insertion and Selection sorts. These algorithms work as follows. **i. Insertion sort-** 1) procedure insertionSort(A: list of sortable items) 2) n = length(A) 3) for i = 1 to n - 1 do 4) j = i 5) while j > 0 and A[j-1] > A[j] do 6) swap(A[j], A[j-1]) 7) j = j - 1 8) end while 9) end for 10)end procedure **ii. Selection sort-** 1) Repeat Steps b and c for i = 0 to n-1 2) CALL SMALLEST(arr, i, n, pos) 3) SWAP arr[i] with arr[pos] 4) [END OF LOOP] 5) EXIT 6) SMALLEST (arr, i, n, pos) 7) [INITIALIZE] SET SMALL = arr[i] 8) [INITIALIZE] SET pos = i

	9) Repeat for j = i+1 to n 10)if (SMALL > arr[j]) 11) SET SMALL = arr[j] 12)SET pos = j 13)[END OF if] 14)[END OF LOOP] RETURN pos
CODE:1 a:	

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

void n()
{
    for (int i = 0; i <= 100; i++)
    {
        printf("%d, %d\n",i,i);
    }
}
void n3()
{
    double s;
    for (double i = 0; i <= 100; i++)
    {
        s=pow(i,3.0);
        printf("%f, %f\n",i,s);
    }
}
void n2n()
{
    double s;
    for (double i = 0; i <= 100; i++)
    {
        s=i*pow(2,i);
        printf("%f, %f\n",i,s);
    }
}
void e_n()
{
    double s;
    for (double i = 0; i <= 100; i++)
    {
        s=exp(i);
        printf("%f, %f\n",i,s);
    }
```

```
}

void p_2n()
{
    double s;
    for (double i = 0; i <= 100; i++)
    {
        s=pow(2,i);
        printf("%f, %f\n",i,s);
    }
}
void p_32n()
{
    double s;
    for (double i = 0; i <= 100; i++)
    {
        s=pow(1.5,i);
        printf("%f, %f\n",i,s);
    }
}
void p_2log()
{
    double s;
    for (double i = 0; i <= 100; i++)
    {
        s=log2(i);
        s=pow(2,s);
        printf("%f, %f\n",i,s);
    }
}
void fact()
{
    double s;
    for (double i = 0; i <= 20; i++)
    {
        s=1;
        for (double j = 1; j <= i; ++j)
        {
            s=s*j;
        }
        printf(" %f\n",s);
    }
}
void loglogn()
{
    double s;
    for (double i = 0; i <= 100; i++)
    {
        s=log2(i);
```

```
        s=log2(s);
        printf("%f, %f\n",i,s);
    }
}
void log2n()
{
    double s;
    for (double i = 0; i <= 100; i++)
    {
        s=log2(i);
        s=pow(s,2);
        printf("%f, %f\n",i,s);
    }
}
void log_2n()
{
    double s;
    for (double i = 0; i <= 100; i++)
    {
        s=log2(i);
        s=pow(s,0.5);
        printf("%f, %f\n",i,s);
    }
}
void main()
{
    n();
    n3();
    n2n();
    e_n();
    p_2n();
    p_32n();
    fact();
    p_2log();
    loglogn();
    log2n();
    log_2n();
}
```

Conclusion-

The gradient of all Logarithmic functions decreases and gradient of all Exponential function increase as n increases and other graphs are linear.

Code:1 b:

```c
#include <stdio.h>
#include<stdlib.h>
#include<time.h>
void main()
{
    int n=0;
    for(int k=0; k<(100000/100); k++)
    {
        n=n+100;
        int num[n];
        int insert[n];
        int select[n];
        int j, min;
        clock_t start_t, end_t;
            double total_t;
        printf("%d\t",n);
        for(int i=0; i<n; i++)
        {
            num[i]=rand() % 10;
            insert[i]=num[i];
            select[i]=num[i];
        }
        start_t = clock();
        for (int i = 1; i < n; i++)
        {
            int a = insert[i];
            j = i - 1;
            while (j >= 0 && insert[j] > a)
            {
                insert[j + 1] = insert[j];
                j = j - 1;
            }
            insert[j + 1] = a;
        }
        end_t = clock();
        total_t = (double)(end_t - start_t) / CLOCKS_PER_SEC;
        printf("%f\t", total_t  );
        start_t = clock();
        for (int i = 0; i < n; i++)
        {
            min = i;
            for (j = i+1; j < n; j++)
            {
              if (select[j] < select[min])
                {
                    min = j;
                }
            }
```

Conclusion-	**I learnt the Insertion Sort and Selection sort algorithm and their time complexities. I also understood how to calculate them and draw similar inferences.**

```
            if(min != i)
            {
                int temp=select[i];
                select[i]=select[min];
                select[min]=temp;
            }
        }
        end_t = clock();
        total_t = (double)(end_t - start_t) / CLOCKS_PER_SEC;
        printf("%f\n", total_t  );
    }
}
```