Name	**Om Doshi**
UID	**2021300030**
Subject	**Data Analysis Algorithm**
Experiment No	**1**

Aim-

1. To implement the various functions e.g. linear, non-linear, quadratic, exponential etc.
2. Experiment on finding the running time of an algorithm.

Algorithm-

1. **Insertion sort-**

```
a. procedure insertionSort(A: list of sortable items)
b.    n = length(A)
c.    for i = 1 to n - 1 do
d.        j = i
e.        while j > 0 and A[j-1] > A[j] do
f.            swap(A[j], A[j-1])
g.            j = j - 1
h.        end while
i.    end for
j. end procedure
```

2. **Selection sort-**

```
a. Repeat Steps b and c for i = 0 to n-1
b. CALL SMALLEST(arr, i, n, pos)
c. SWAP arr[i] with arr[pos]
d. [END OF LOOP]
e. EXIT

f. SMALLEST (arr, i, n, pos)
g.  [INITIALIZE] SET SMALL = arr[i]
h.  [INITIALIZE] SET pos = i
i. Repeat for j = i+1 to n
j. if (SMALL > arr[j])
k.     SET SMALL = arr[j]
l. SET pos = j
m. [END OF if]
n. [END OF LOOP]
o. RETURN pos
```

Code-

1. 1A-

```
#include<stdio.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 p_2n()
{
        double s;
        for (double i = 0; i <= 100; ++i)
        {
                s=pow(2,i);
                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 en()
{
        double s;
        for (double i = 0; i <= 100; ++i)
        {
                s=exp(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 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 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 main()
{
        n();
        n3();
        p_2n();
        n2n();
        en();
        p_32n();
        p_2log();
        loglogn();
        log2n();
        log_2n();
        fact();
}
```

2. 1B-

```
#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;
                 }
                        }
```

```
                    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  );
            }
    }
```

Conclusion-

Thus I have understood the Insertion and Selection sort algorithm and their time complexities. I also understood how to calculate them and draw similar inferences.