Name	SMRUTI SONEKAR
UID no.	2021700064
Experiment No.	1A

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

Program 1

PROBLEM STATEMENT :

$(\frac{3}{2})^n$	n^3	$\lg^2 n$	$\lg(n!)$	2^{2^n}	$n^{1/\lg n}$
$\ln \ln n$	$\lg\ \ n$	$n \cdot 2^n$	$n^{\lg \lg n}$	$\ln n$	$2^{\lg\ n}$
$2^{\lg n}$	$(\lg n)^{\lg n}$	e^n	$(\lg n)!$	$(\sqrt{2})^{\lg n}$	$\sqrt{\lg n}$
$\lg\ (\lg n)$	$2^{\sqrt{2 \lg n}}$	n	2^n	$n \lg n$	$2^{2^{n+1}}$

Note – *lg* denotes for log_2 and *le* denotes log_e

ALGORITHM/ THEORY:

Step 1:We start the program by initializing/declaring all the functions in double so it will return value in double.

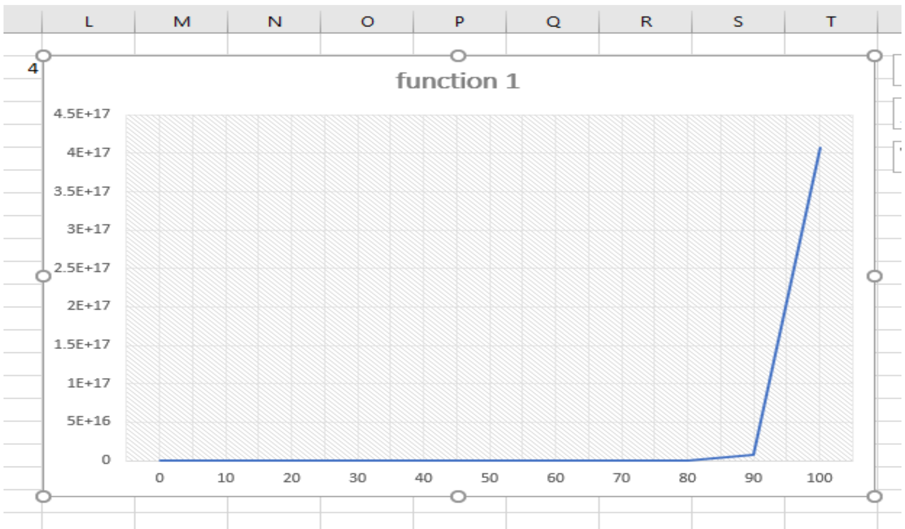
Step 2:In the main function , an array was initialized from 0,10,20..100 as we will use this input for 10 functions .

Step 3:Initialized t1,t2..t10 as we will use this to call our functions and functions will return some values which will be stored in t1,t2…. Variables.

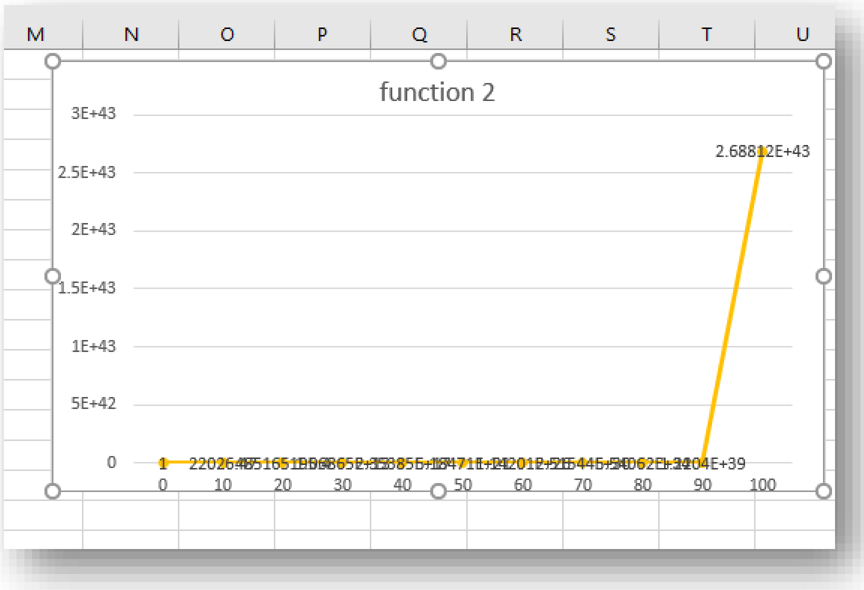
Step 4:All the functions are created according and math.h module is used for some predefined functions like sqrt,pow,log etc.

1] Function 1= (3/2)^n graph.

It shows a rapid spike between 90 to100 input value,before that 0 to 80 it almost linear graph , as input value increases result increases.

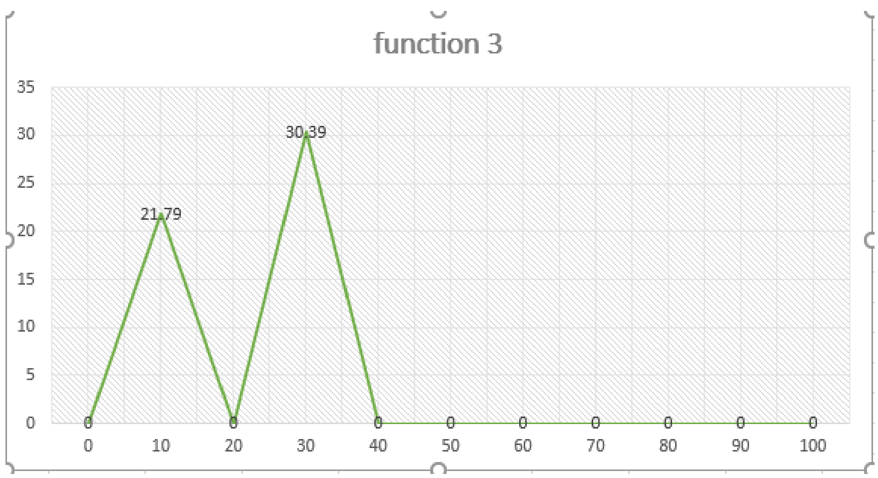


2] Function 2: e^n graph , exponential graph.In this graph line there is rapid growth between 90 to 100 and and from 0 to 80 it is increases in value but in small increment.



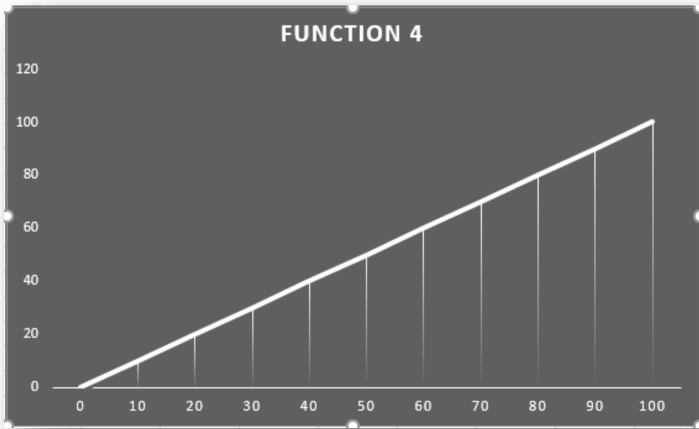
3] Function 3= log2(n!) ,logarithmic graph

In this graph , the line has spiked at the value of 10 which is 21.79 and value of 30 at 30.39 and it is infinity at all other values.It is 0 at 0 value.

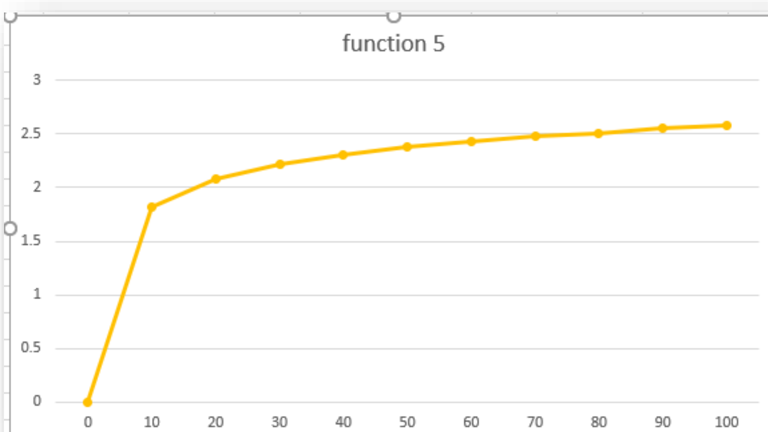


4] Function 4= 2^(log2(n)).

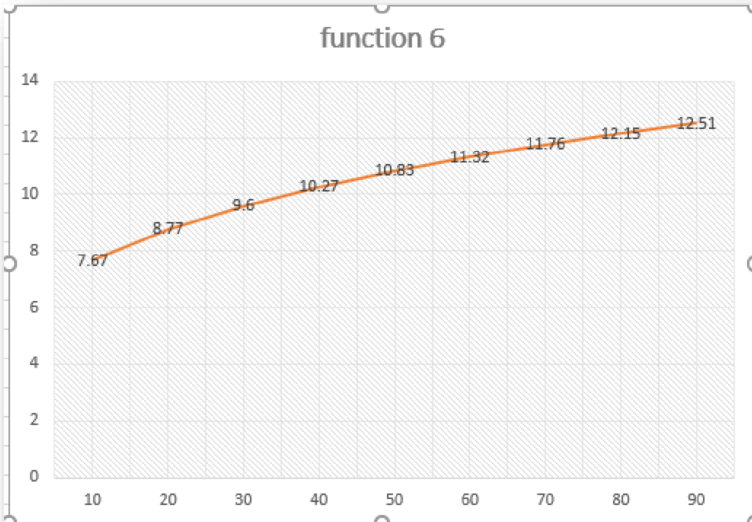
It a straight line which indicates as linear graph, as value = result.Ex: y=x , slope of line is m=1;c=0;



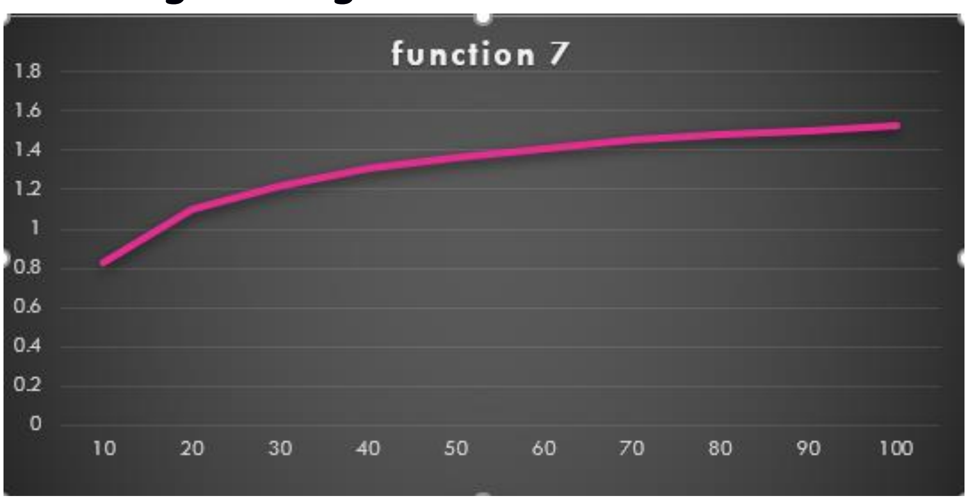
5] Function 5= Square root of log2(n).All the output/results are less than 3.From 0 to 10 there is straight line which indicates that it has sharply increased and from 10 to 100 there is slow growth as the line proceeds.



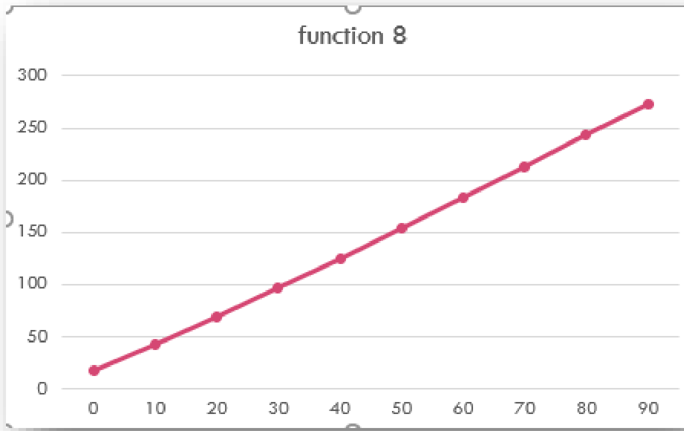
6] Function 6 = 2^(2*log2(n)) graph .As the input values increases, result also increases .The line has gradual increase.



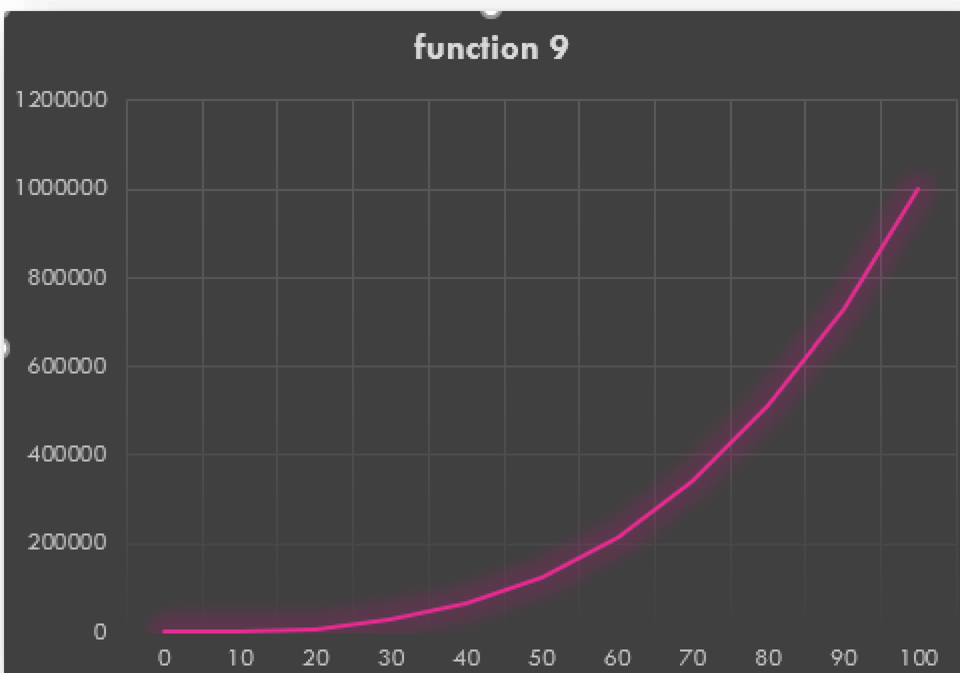
7] Function 7=ln(ln(n)) natural logarithmic graph .Its a smooth gradual growth as line shows.



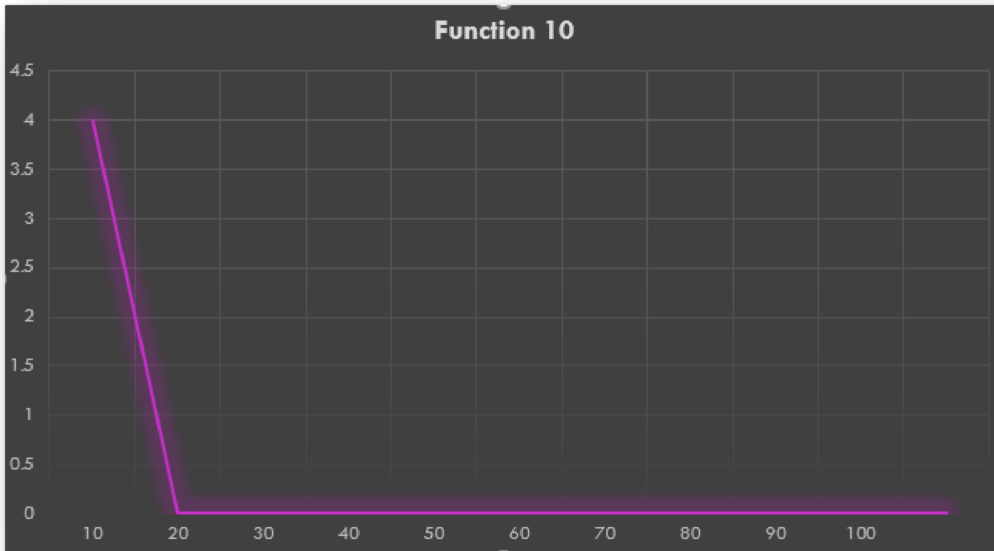
8] Function 8: n*(log2(log2(n))) function graph.It is linear graph /straight line which indicates input is directly proportional to the output/ result.



9] Function 9=n^3 cubic graph.It is first quandrant A cubic function is a polynomial function of degree 3. So the graph of a cube function may have a maximum of 3 roots.



10] Function 10= 2^(2^(n+1)) graph.

	In this graph line decreases as value input increases, And from 20 to 100 result is infinity as it's a huge value. 
PROGRAM:	(see code below)

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

double f1(double x,double y);
double f2( double x);
double f3(double x);
double f4(double x);
double f5(double x);
double f6(double x);
double f7(double x);
double f8(double x);
double f9(double x);
double f10(double x);


void main()
{

   double n[]={0,10,20,30,40,50,60,70,80,90,100};
   double t1,t2,t3,t4,t5,t6,t7,t8,t9,t10;
   int i;
```

```
 printf("\n 1] F1((3/2)^n) \n");
for(i=0;i<11;i++)
{

     t1= f1( 1.5, n[i]);
   printf("\n%0.1lf= %0.2lf\n" ,n[i],t1);
}


   printf("\n 2] F2[e^n]  \n");
    for(i=0;i<11;i++)
    {
        t2= f2(n[i]);
         printf("\n%0.1lf= %0.2lf\n" ,n[i],t2);
    }

    printf("\n 3] F3[lg(n!)]\n");
     for(i=0;i<11;i++)
    {
        t3=f3(n[i]);
        printf("\n%0.1lf= %0.2lf\n" ,n[i],t3);
    }

   printf("\n 4] F4[2^(lg n)]  \n");
    for(i=0;i<11;i++)
    {
        t4=f4(n[i]);
        printf("\n%0.1lf= %0.2lf\n" ,n[i],t4);
    }
    printf("\n 5] F5[(lg n)^0.5]  \n");
    for(i=0;i<11;i++)
    {
        t5=f5(n[i]);
        printf("\n%0.1lf= %0.2lf\n" ,n[i],t5);
    }

     printf("\n 6]  F6[2^((2*log2(x))^0.5)]  \n");
    for(i=0;i<11;i++)
    {
        t6=f6(n[i]);
        printf("\n%0.1lf= %0.2lf\n" ,n[i],t6);
    }
```

```c
        printf("\n 7] F7[ln(ln n)] \n");
        for(i=0;i<11;i++)
        {
            t7=f7(n[i]);
            printf("\n%0.1lf= %0.2lf\n" ,n[i],t7);
        }

        printf("\n 8] F8[n*lg(lg n)] \n");
        for(i=0;i<11;i++)
        {
            t8=f8(n[i]);
            printf("\n%0.1lf= %0.2lf\n" ,n[i],t8);
        }

        printf("\n 9] F9[n^3] \n");
        for(i=0;i<11;i++)
        {
            t9=f9(n[i]);
            printf("\n%0.1lf= %0.2lf\n" ,n[i],t9);
        }

        printf("\n 10] F10[2^(2^n+1)] \n");
        for(i=0;i<11;i++)
        {
            t10=f10(n[i]);
            printf("\n%0.1lf= %0.2lf\n" ,n[i],t10);
        }
}


double f1(double x,double y)
{
    return pow(x,y);    //(3/2)^n
}

double f2(double x)
{
    return exp(x);   //e^n
}

double f3(double x)
{
    int i,fact=1;
```

```
    for(i=1;i<=x;i++)
    {
        fact=fact*i;   //lg(n!)
    }
    return log2(fact);
}

double f4(double x)
{
    double res=log2(x);
    return pow(2,res);          //2^(lg n)
}

double f5(double x)
{
    double res=log2(x);
    return sqrt(res);       //(lg n)^0.5
}

double f6(double x)
{
    double res=sqrt(2*log2(x));      //2^((2*log2(x))^0.5)
    return pow(2,res);
}

double f7(double x)
{
    return log(log(x));   //ln(ln n)
}

double f8(double x)
{
   return x*(log2(log2(x)));     //n*lg(lg n)

}

double f9(double x)
{

    return pow(x,3);              //n^3
}

double f10(double x)
{
```

	`` double res=pow(2,x+1); return pow(2,res); //2^(2^n+1) } ```

RESULT:	<pre>PS D:\c_programming\mudir\daa> gcc exp1A.c PS D:\c_programming\mudir\daa> .\a.exe 1] F1((3/2)^n) 0.0= 1.00 10.0= 57.67 20.0= 3325.26 30.0= 191751.06 40.0= 11057332.32 50.0= 637621500.21 60.0= 36768468716.93 70.0= 2120255184830.25 80.0= 122264598055704.64 90.0= 7050392822843069.00 100.0= 406561177535215230.00 2] F2[e^n] 0.0= 1.00 10.0= 22026.47 20.0= 485165195.41 30.0= 10686474581524.46</pre>

```
40.0= 235385266837020000.00

50.0= 5184705528587072000000.00

60.0= 114200738981568420000000000.00

70.0= 2515438670919166900000000000000.00

80.0= 55406223843935098000000000000000000.00

90.0= 1220403294317840800000000000000000000000.00

100.0= 26881171418161356000000000000000000000000000.00

 3] F3[lg(n!)]

0.0= 0.00

10.0= 21.79

20.0= -1.#J

30.0= 30.39

40.0= -1.#J

50.0= -1.#J

60.0= -1.#J

70.0= -1.#J

80.0= -1.#J

90.0= -1.#J
```

```
 4] F4[2^(lg n)]

0.0= 0.00

10.0= 10.00

20.0= 20.00

30.0= 30.00

40.0= 40.00

50.0= 50.00

60.0= 60.00

70.0= 70.00

80.0= 80.00

90.0= 90.00

100.0= 100.00

 5] F5[(lg n)^0.5]

0.0= -1.#J

10.0= 1.82

20.0= 2.08

30.0= 2.22

40.0= 2.31
```

```
40.0= 9.60
50.0= 10.27
60.0= 10.83
70.0= 11.32
80.0= 11.76
90.0= 12.15
100.0= 12.51
 7] F7[ln(ln n)]
0.0= -1.#J
10.0= 0.83
20.0= 1.10
30.0= 1.22
40.0= 1.31
50.0= 1.36
60.0= 1.41
70.0= 1.45
80.0= 1.48
90.0= 1.50

 8] F8[n*lg(lg n)]
0.0= -1.#J
10.0= 17.32
20.0= 42.23
30.0= 68.84
40.0= 96.48
50.0= 124.83
60.0= 153.74
70.0= 183.10
80.0= 212.83
90.0= 242.88
100.0= 273.20
 9] F9[n^3]
0.0= 0.00
10.0= 1000.00
20.0= 8000.00
30.0= 27000.00
40.0= 64000.00
50.0= 125000.00

10.0= 1000.00
20.0= 8000.00
30.0= 27000.00
40.0= 64000.00
50.0= 125000.00
60.0= 216000.00
70.0= 343000.00
80.0= 512000.00
90.0= 729000.00
100.0= 1000000.00
 10] F10[2^(2^n+1)]
0.0= 4.00
10.0= 1.#J
20.0= 1.#J
30.0= 1.#J
40.0= 1.#J
50.0= 1.#J
60.0= 1.#J
70.0= 1.#J
```

CONCLUSION:	In this experiment ,through the graphs of functions and output of every function for every value the analysis of each function is clear .There was exponential graph, logarithmic graphs, linear graph (for complicated function) .Through line graph difference between one function and another function was visible and concept of analysis of algorithm was clearly understood.