NAME:	Om Bhutki
UID:	2022301003
SUBJECT	DAA
EXPERIMENT NO :	1A
AIM:	To implement the various functions e.g. linear, non-linear, quadratic, exponential etc.
ALGORITHM	Step 1: Start. Step 2: Declare the variables which are required to perform operations on the functions. Step 3: Start the loop which starts from 0 th number to 100 th number. Step 4: i. perform the operation: 3/2^n using pow() ii. Print the result. Step 5: i. Perform the operation: n^3 using simple multiplication ii. Print the result. Step 6: i. Perform the operation: n.lg(n) using in built log function in math.h ii. Print the result. Step 7: i. Perform the operation: lg(n) using in built log function in math.h ii. Print the result. Step 8: i. Perform the operation: 2^lg(n) using in built log function in math.h and pow() ii. Print the result. Step 9: i. Perform the operation lg(lg(n)) using in built log function in math.h

ii. Print the result.
Step 10: i. Perform the operation:
lg(n)^2 using in built log function in math.h and pow()
ii. Print the result.
Step 11: i. Perform the operation:
n
ii. Print the result.
Step 12: i. Perform the operation:
2^n using pow()
ii. Print the result.
Step 13: i. Perform the operation:
n.2^n using pow()
ii. Print the result.
Step 14: End the loop
Step 15: End.

Algorithm for Factorial of numbers from 1 to 20:
Step 1: Start.
Step 2: Declare the variables n, fact
Step 3: Initialize the values n = 20 and fact = 1.
Step 4: Start the loop from 1 to n
Step 5: calculate, fact = fact *i
Step 6: print the value of fact
Step 7: End.

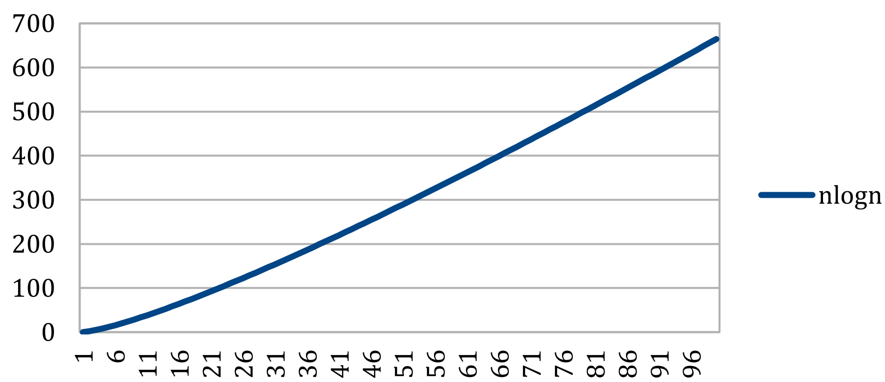
PROGRAM:

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

int main()
{
    int i;
    long double a,b,c,d,g,k;
    float e,f,h;
```

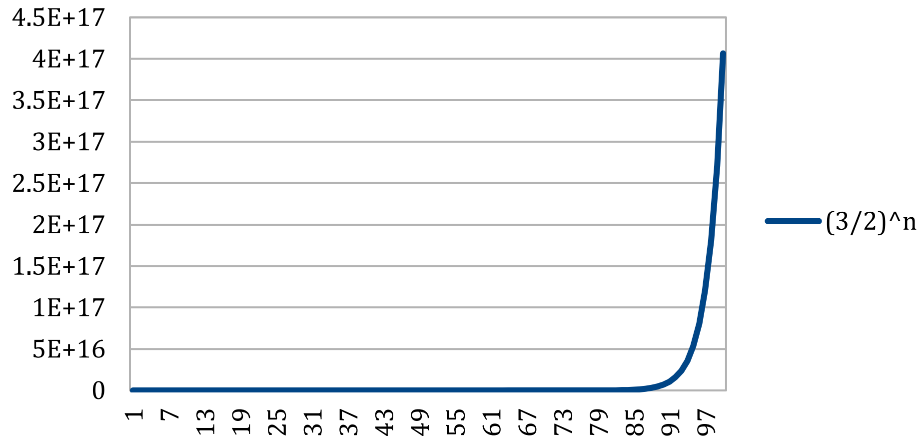
```
     for(i = 1;i<=100;i++){
        a = pow(1.5,i);
        printf("%f\n",a);
          b = i * i * i;
        printf("%f\n",b);
          printf("%d\n",i);
          c = pow(2,i);
          printf("%f\n",c);
          d = i*log2(i);
          printf("%Lf\n",d);
          e = pow(2,log2(i));
          printf("%f\n",e);
           f = log2(i);
           printf("%f\n",f);
          g = i*pow(2,i);
           printf("%Lf\n",g);
           h = log10(log10(i));
           printf("%f\n",h);
             k = pow(log2(i),2);
             printf("%Lf\n",k);



     }

    return 0;
}

#include<stdio.h>
void fact(int num){
    int i;
    long f=1;
```

```
  for(i=1;i<=num;i++)
      f=f*i;

  printf("%d %ld\n",num,f);
}

int main(){
  int i;
  for(i = 1; i<=20;i++){
      fact(i);
  }
  return 0;
}
```

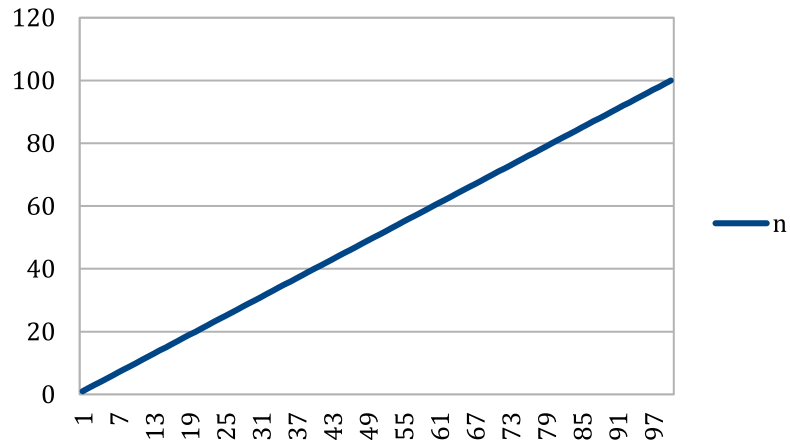
Graph:



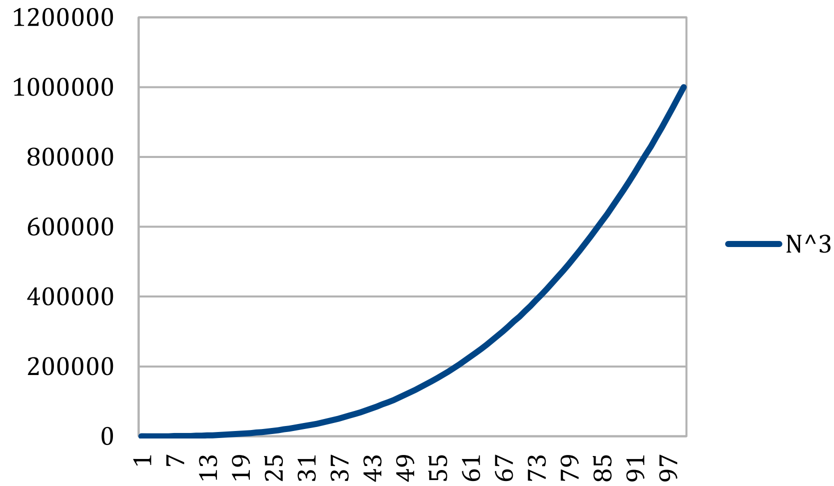
Inference: We see that there is a slow increase between 0 and 1 and after that there is a uniform increase till the 100th value.



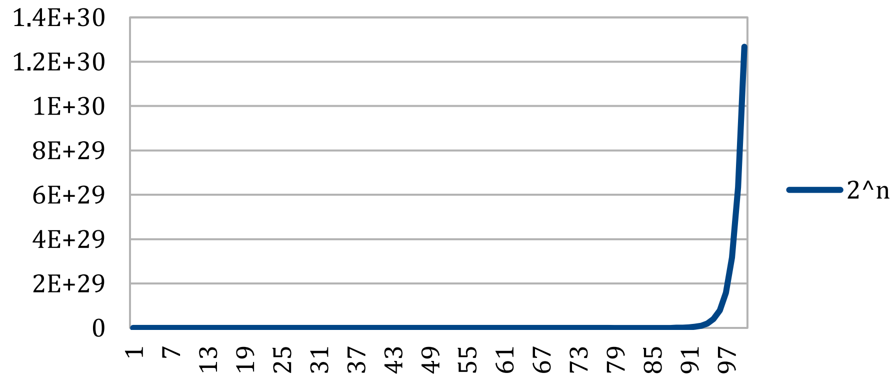
Inference: The graph doesn't have a huge increase for a few iterations after which the values suddenly rises rapidly, going over millions.



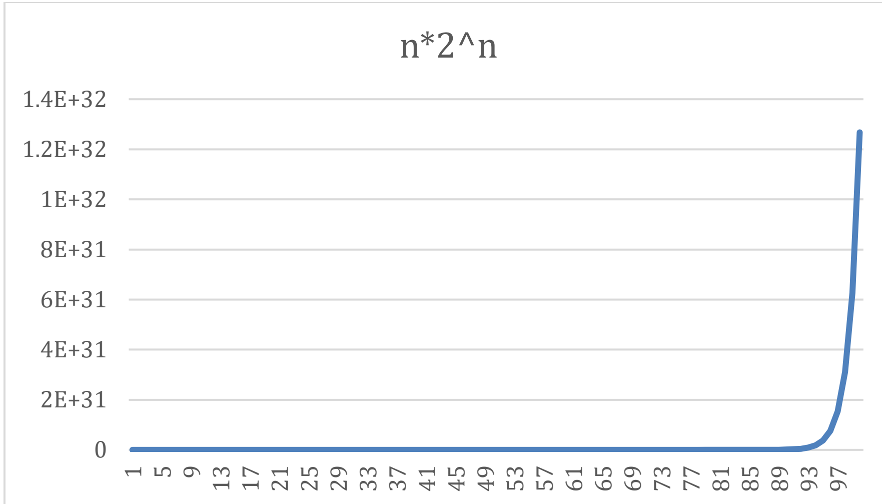
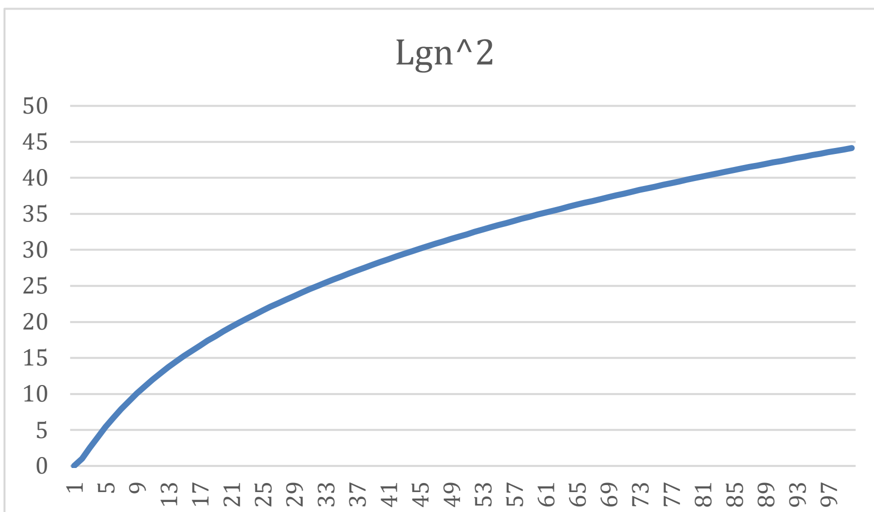
Inference: There is a linear increase in the graph since we are simply printing values from 1 to 100.

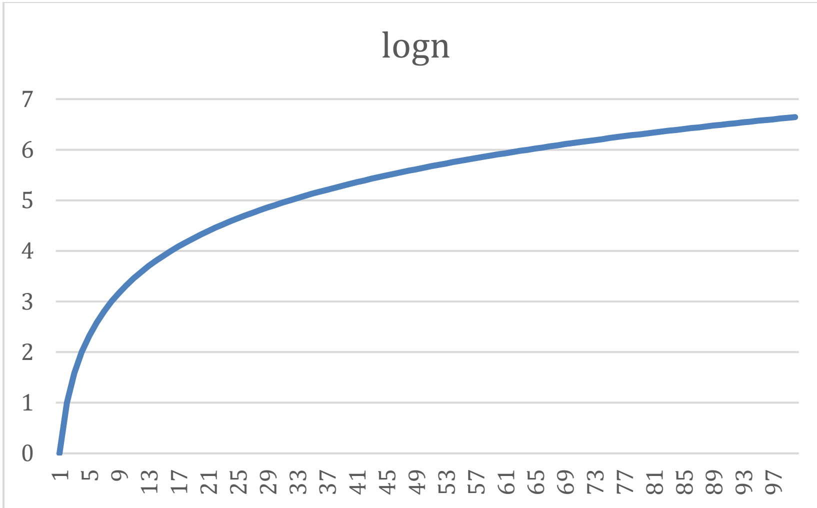
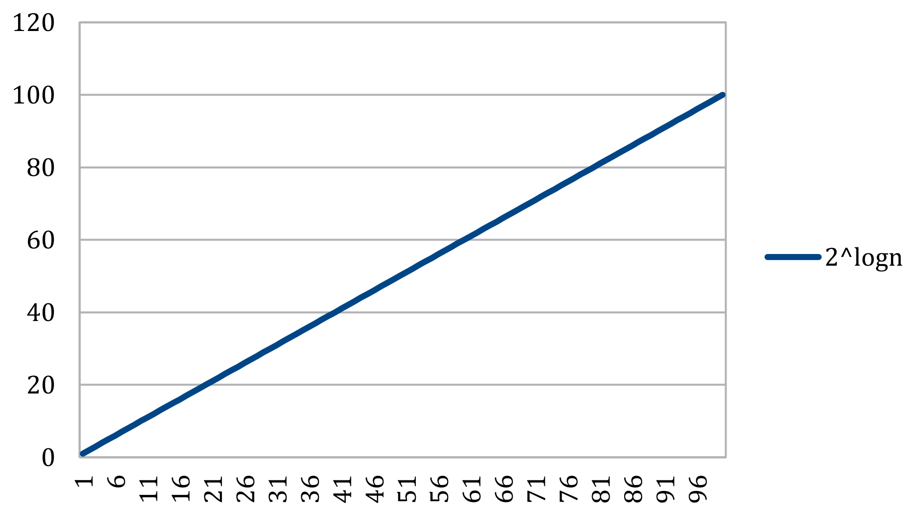


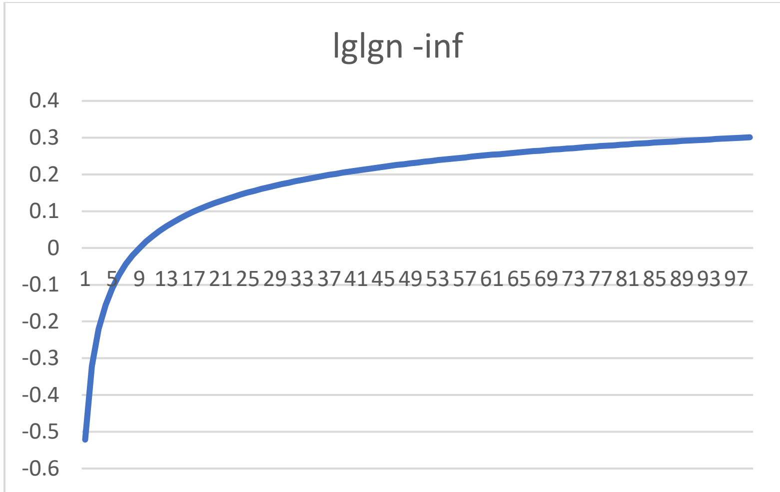
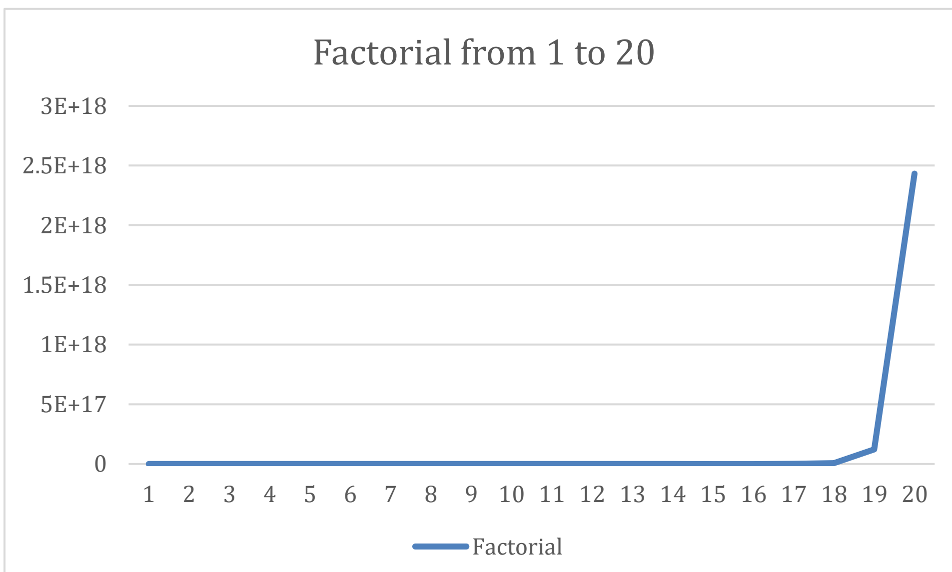
Inference: For N^3 there is a gradual slope where are similar till 19 after which they increase gradually.



Inference: For 2^n the graph doesn't have a huge increase for a few iterations after which the values suddenly rises rapidly, going over millions.

	 **Inference:** For n*2^n the graph doesn't have a huge increase for a few iterations after which the values suddenly rises rapidly, going over millions.  **Inference:** We can see a slope where the angle of slop is gradually decreasing.

	 **Inference:** We can see a slope where the angle of slop is gradually decreasing lesser than lgn^2  **Inference:** We can see a slope which is increasing with the same angle i.e a graph which is linear.

	 **Inference:** Since log(log(1)) is -inf the graph starts from the negative quadrant and makes its way up after which we observe normal logarithmic graph behaviour which is linear increase over a few values followed by slow increase.  **Inference:** We see that there is not much increase around the starting numbers but the graph takes a sudden increase around the ending numbers basically going to really high numbers.

Conclusion:	Thus, after running 10 functions on numbers from 1 to 100 We conclude that: -Functions with power increase rapidly after some slow growth. -Functions with log have linear increase for some values after which we observe slow growth.