Name	**Nishant Dinesh Satere**
Uid No.	**2022701009**
Branch	**CSE-DS**
Batch	**D**
Subject	**DAA-Lab**
Experiment No.	**1**

PROGRAM/SOURCE CODE:

Experiment1.c

C: > Users > Admin > Desktop > SPIT > Sem4 > Praticals > DDA > Experiment1.c > function3()

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

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

//(3/2)^n
void function1()
{
    for(int n = 0; n <= 100; n = n+10)
    {
        printf("%d : %f\n", n , pow(1.5,n));
    }
}

//n^3
void function2()
{
    for(int n = 0; n <= 100; n=n+10)
    {
        printf("%d : %d\n",n,n*n*n);
    }
}

//2^(2^n)
void function3()
{
    for(int n = 0; n <= 100; n = n+10)
    {
        printf("%u : %f\n",n, 2*(pow(2,n)));
    }
}

//log2(n)
void function4()
{
    for(int n = 0; n <= 100; n=n+10)
    {
```

Experiment1.c

C: > Users > Admin > Desktop > SPIT > Sem4 > Praticals > DDA > Experiment1.c > function3()

```c
        printf("%d : %f\n", n ,log2(n));
    }
}

//2^(log2(n))
void function5()
{
    for(int n = 0; n <= 100; n= n+10)
    {
        printf("%d : %f\n",n,pow(2,(log2(n))));
    }
}

//n
void function6()
{
    for(int n = 0; n <= 100; n = n+10)
    {
        printf("%d : %d\n",n,n);
    }
}

//2^n
void function7()
{
    for (int n = 0; n <= 100; n = n+10)
    {
        printf("%d : %f\n",n,pow(2,n));
    }

}

//nlog2(n)
void function8()
{
    for(int n = 0; n <= 100; n = n+10)
    {
```

Experiment1.c

C: > Users > Admin > Desktop > SPIT > Sem4 > Praticals > DDA > Experiment1.c > function7()

```c
        printf("%d : %f\n",n,n*log2(n));
    }
}

//2^((2^n)+1)
void function9()
{
    for(int n =0; n <= 100; n = n+10)
    {
        printf("%d : %f\n",n,pow(2,pow(2,n+1)));
    }
}

//lnln(n)
void function10()
{
    for(int n = 0; n <= 100; n = n+10)
    {
        printf("%d : %f\n",n,log(log(n)));
    }
}

//n!
void function11()
{
    for(int n = 0; n <= 20; n = n+2)
    {
        double fact = 1;
        for(int i = 1; i <= n; i++)
        {
            fact = fact*i;
        }
        printf("%d : %f\n",n,fact);
    }

}
int main()
```

Experiment1.c

C: > Users > Admin > Desktop > SPIT > Sem4 > Praticals > DDA > Experiment1.c > function7()

```c
{
    printf("Press 1 : Function1 (3/2)*n\n");
    printf("Press 2 : Function2 n^3\n");
    printf("Press 3 : Function3 2^(2^n)\n");
    printf("Press 4 : Function4 log2(n)\n");
    printf("Press 5 : Function5 2^(log2(n))\n");
    printf("Press 6 : Function6 n\n");
    printf("Press 7 : Function7 2^n\n");
    printf("Press 8 : Function8 nlog2(n)\n");
    printf("Press 9 : Function9 2^((2^n)+1)\n");
    printf("Press 10 : Function10 lnln(n)\n");
    printf("Press 11 : Function11 n!\n");
    printf("Press 12 : Exit\n");

    int option;
    while (option != 12)
    {
        printf("Enter your choice : ");
        scanf("%d" ,&option);
        switch (option)
        {
        case 1:
        printf("Function1 (3/2)*n : \n");
        function1();
            break;

        case 2:
        printf("Function2 n^3 : \n");
        function2();
            break;

        case 3:
        printf("Function3 2^(2^n) : \n");
        function3();
            break;

        case 4:
```

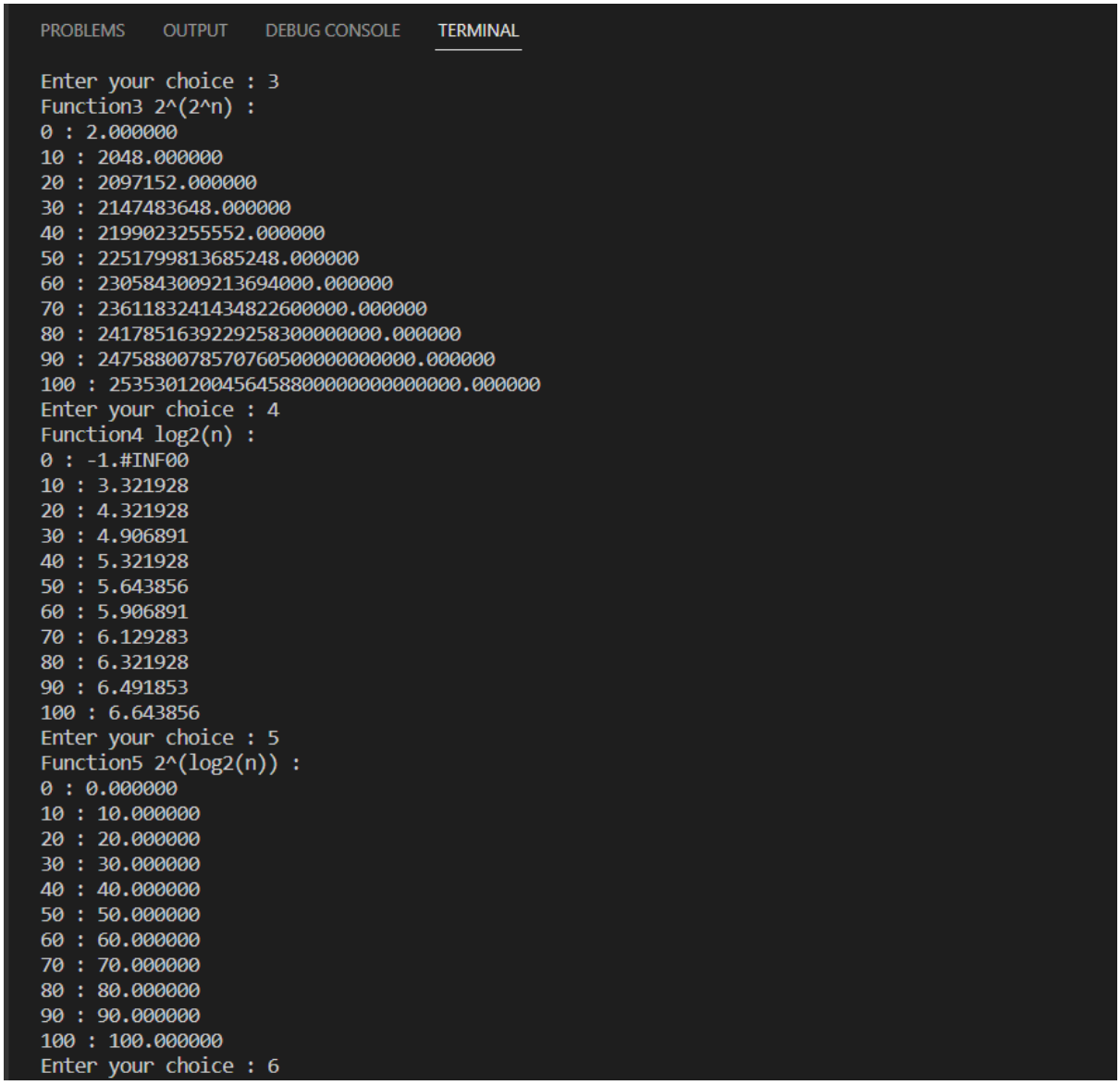
Experiment1.c

C: > Users > Admin > Desktop > SPIT > Sem4 > Praticals > DDA > Experiment1.c > function7()

```c
            printf("Function4 log2(n) : \n");
            function4();
                break;

            case 5:
            printf("Function5 2^(log2(n)) : \n");
            function5();
                break;

            case 6:
            printf("Function6 n : \n");
            function6();
                break;

            case 7:
            printf("Function7 2^n : \n");
            function7();
                break;

            case 8:
            printf("Function8 nlog2(n) : \n");
            function8();
                break;

            case 9:
            printf("Function9 2^((2^n)+1) : \n");
            function9();
                break;

            case 10:
            printf("Function10 lnln(n) : \n");
            function10();
                break;

            case 11:
            printf("Function11 n! : \n");
            function11();
                break;
        }
    }
}
```

OUTPUT:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\Users\Admin\Desktop\SPIT\sem4\Praticals\DDA> gcc Experiment1.c
PS C:\Users\Admin\Desktop\SPIT\sem4\Praticals\DDA> ./a.exe
Press 1 : Function1 (3/2)*n
Press 2 : Function2 n^3
Press 3 : Function3 2^(2^n)
Press 4 : Function4 log2(n)
Press 5 : Function5 2^(log2(n))
Press 6 : Function6 n
Press 7 : Function7 2^n
Press 8 : Function8 nlog2(n)
Press 9 : Function9 2^((2^n)+1)
Press 10 : Function10 lnln(n)
Press 11 : Function11 n!
Press 12 : Exit
Enter your choice : 1
Function1 (3/2)*n :
0 : 1.000000
10 : 57.665039
20 : 3325.256730
30 : 191751.059233
40 : 11057332.320940
50 : 637621500.214050
60 : 36768468716.933022
70 : 2120255184830.252000
80 : 122264598055704.640000
90 : 7050392822843069.000000
100 : 406561177535215230.000000
Enter your choice : 2
Function2 n^3 :
0 : 0
10 : 1000
20 : 8000
30 : 27000
40 : 64000
50 : 125000
60 : 216000
70 : 343000
80 : 512000
90 : 729000
100 : 1000000
Enter your choice :
```


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
Enter your choice : 3
Function3 2^(2^n) :
0 : 2.000000
10 : 2048.000000
20 : 2097152.000000
30 : 2147483648.000000
40 : 2199023255552.000000
50 : 2251799813685248.000000
60 : 2305843009213694000.000000
70 : 2361183241434822600000.000000
80 : 2417851639229258300000000.000000
90 : 2475880078570760500000000000.000000
100 : 2535301200456458800000000000000000.000000
Enter your choice : 4
Function4 log2(n) :
0 : -1.#INF00
10 : 3.321928
20 : 4.321928
30 : 4.906891
40 : 5.321928
50 : 5.643856
60 : 5.906891
70 : 6.129283
80 : 6.321928
90 : 6.491853
100 : 6.643856
Enter your choice : 5
Function5 2^(log2(n)) :
0 : 0.000000
10 : 10.000000
20 : 20.000000
30 : 30.000000
40 : 40.000000
50 : 50.000000
60 : 60.000000
70 : 70.000000
80 : 80.000000
90 : 90.000000
100 : 100.000000
Enter your choice : 6

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

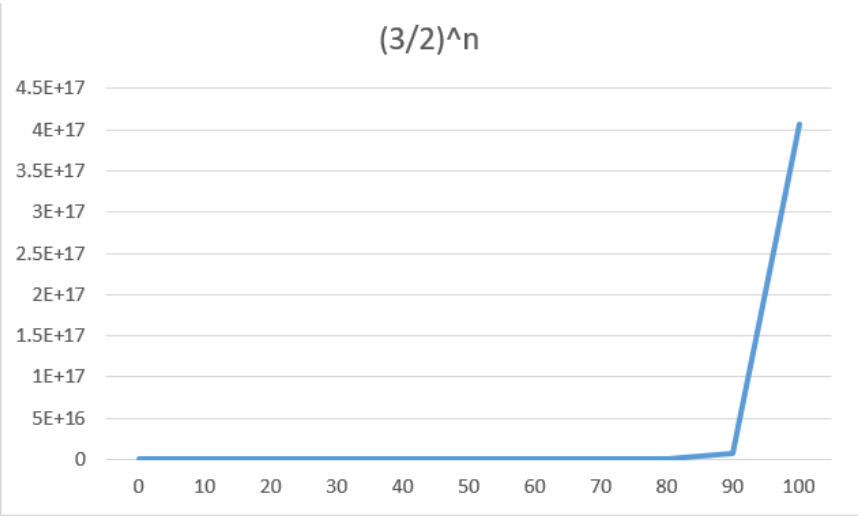
```
Enter your choice : 6
Function6 n :
0 : 0
10 : 10
20 : 20
30 : 30
40 : 40
50 : 50
60 : 60
70 : 70
80 : 80
90 : 90
100 : 100
Enter your choice : 7
Function7 2^n :
0 : 1.000000
10 : 1024.000000
20 : 1048576.000000
30 : 1073741824.000000
40 : 1099511627776.000000
50 : 1125899906842624.000000
60 : 1152921504606847000.000000
70 : 1180591620717411300000.000000
80 : 1208925819614629200000000.000000
90 : 1237940039285380300000000000.000000
100 : 1267650600228229400000000000000.000000
Enter your choice : 8
Function8 nlog2(n) :
0 : -1.#IND00
10 : 33.219281
20 : 86.438562
30 : 147.206718
40 : 212.877124
50 : 282.192809
60 : 354.413436
70 : 429.049811
80 : 505.754248
90 : 584.266779
100 : 664.385619
Enter your choice : 9
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
70 : 429.049811
80 : 505.754248
90 : 584.266779
100 : 664.385619
Enter your choice : 9
Function9 2^((2^n)+1) :
0 : 4.000000
10 : 1.#INF00
20 : 1.#INF00
30 : 1.#INF00
40 : 1.#INF00
50 : 1.#INF00
60 : 1.#INF00
70 : 1.#INF00
80 : 1.#INF00
90 : 1.#INF00
100 : 1.#INF00
Enter your choice : 10
Function10 lnln(n) :
0 : -1.#IND00
10 : 0.834032
20 : 1.097189
30 : 1.224128
40 : 1.305323
50 : 1.364055
60 : 1.409607
70 : 1.446565
80 : 1.477511
90 : 1.504035
100 : 1.527180
Enter your choice : 11
Function11 n! :
0 : 1.000000
2 : 2.000000
4 : 24.000000
6 : 720.000000
8 : 40320.000000
10 : 3628800.000000
12 : 479001600.000000
14 : 87178291200.000000
16 : 20922789888000.000000
18 : 6402373705728000.000000
20 : 2432902008176640000.000000
Enter your choice : 12
PS C:\Users\Admin\Desktop\SPIT\sem4\Praticals\DDA>
```

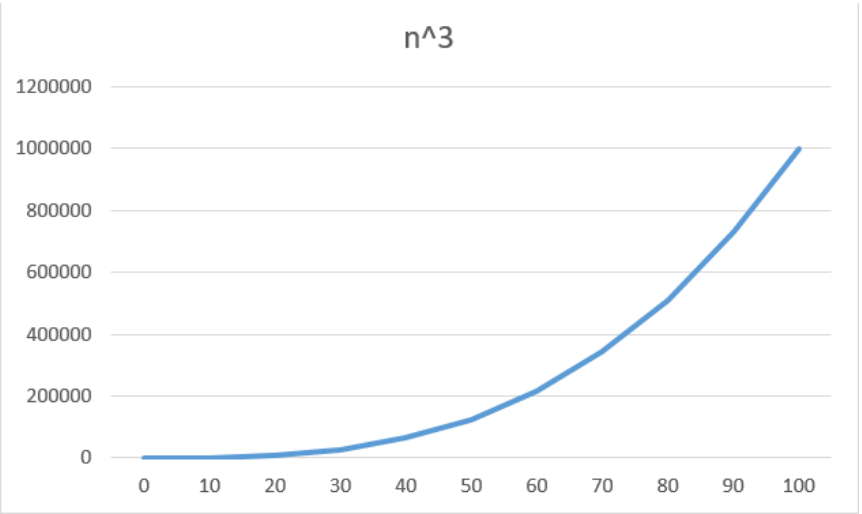
1. (3/2)^n

n	(3/2)^n
0	1
10	57.665039
20	3325.25673
30	191751.0592
40	11057332.32
50	637621500.2
60	36768468717
70	2.12026E+12
80	1.22265E+14
90	7.05039E+15
100	4.06561E+17



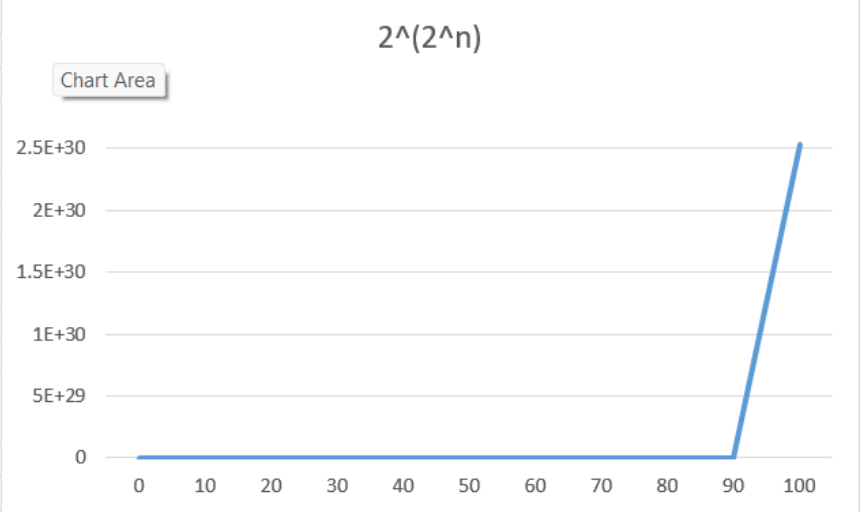
2. n^3

n	n^3
0	0
10	1000
20	8000
30	27000
40	64000
50	125000
60	216000
70	343000
80	512000
90	729000
100	1000000



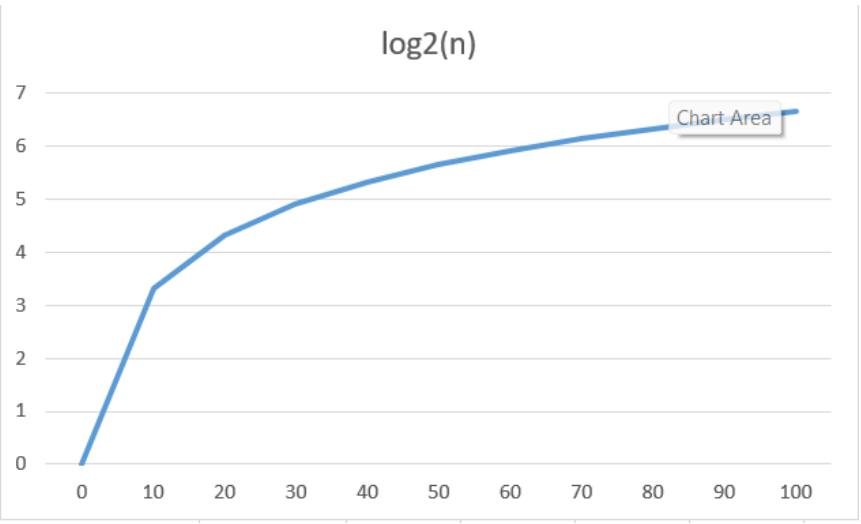
3. 2^(2^n)

n	2^(2^n)
0	2
10	2048
20	2097152
30	2147483648
40	2.19902E+12
50	2.2518E+15
60	2.30584E+18
70	2.36118E+21
80	2.41785E+24
90	2.47588E+27
100	2.5353E+30



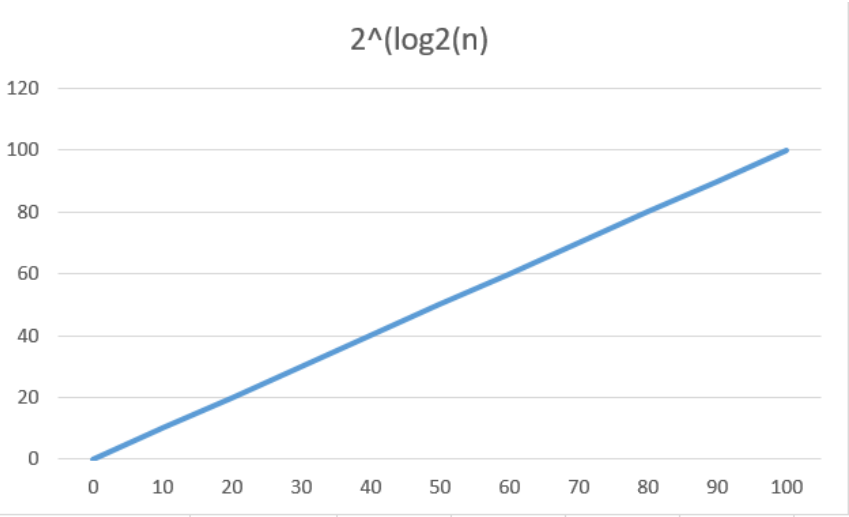
4. Log2(n)

n	log2(n)
0	-1.#INF00
10	3.321928
20	4.321928
30	4.906891
40	5.321928
50	5.643856
60	5.906891
70	6.129283
80	6.321928
90	6.491853
100	6.643856



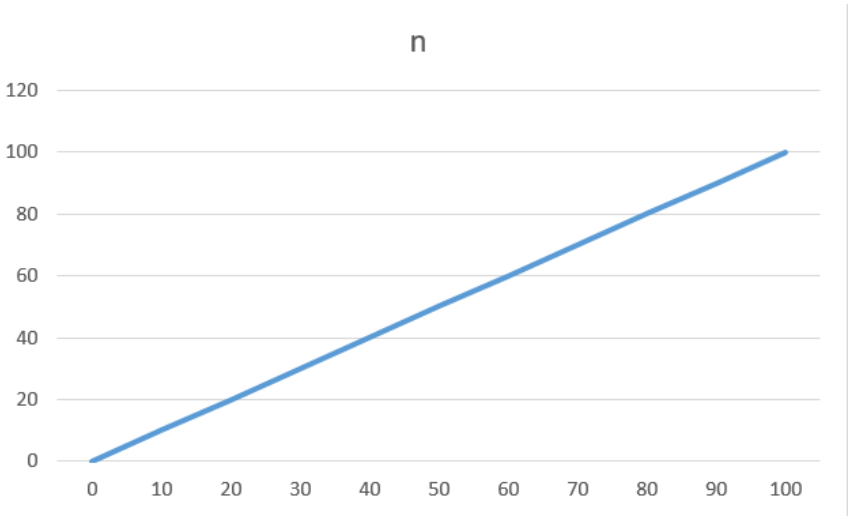
5. 2^(log2(n))

n	2^(log2(n)
0	0
10	10
20	20
30	30
40	40
50	50
60	60
70	70
80	80
90	90
100	100



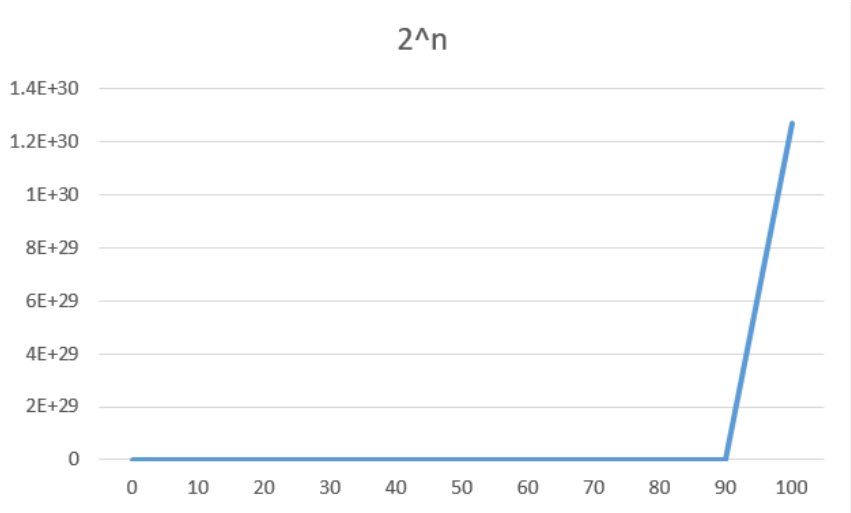
6. n

n	n
0	0
10	10
20	20
30	30
40	40
50	50
60	60
70	70
80	80
90	90
100	100



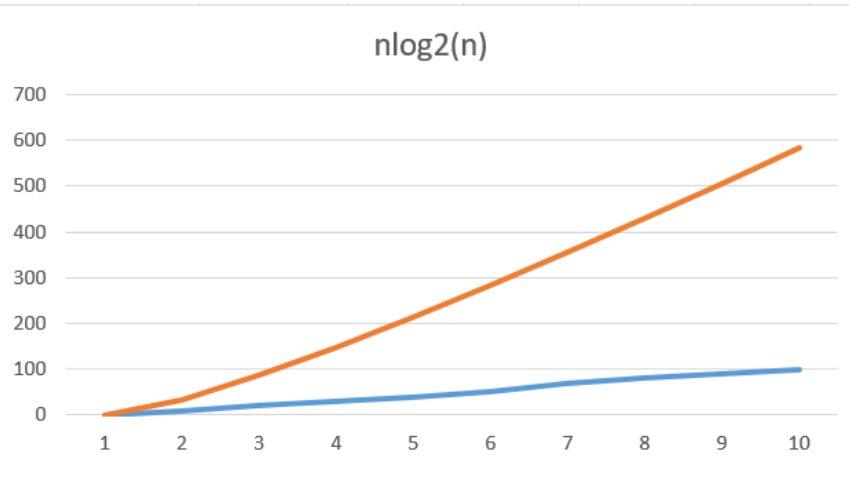
7. 2^n

n	2^n
0	1
10	1024
20	1048576
30	1073741824
40	1.09951E+12
50	1.1259E+15
60	1.15292E+18
70	1.18059E+21
80	1.20893E+24
90	1.23794E+27
100	1.26765E+30



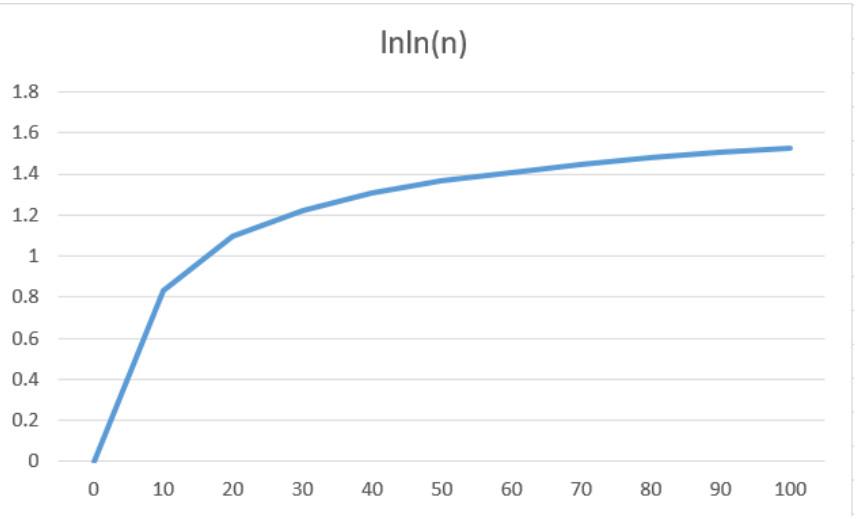
8. nlog2(n)

n	nlog2(n)
0	nan
10	33.219281
20	86.438562
30	147.206718
40	212.877124
50	282.192809
70	354.413436
80	429.049811
90	505.754248
100	584.266779



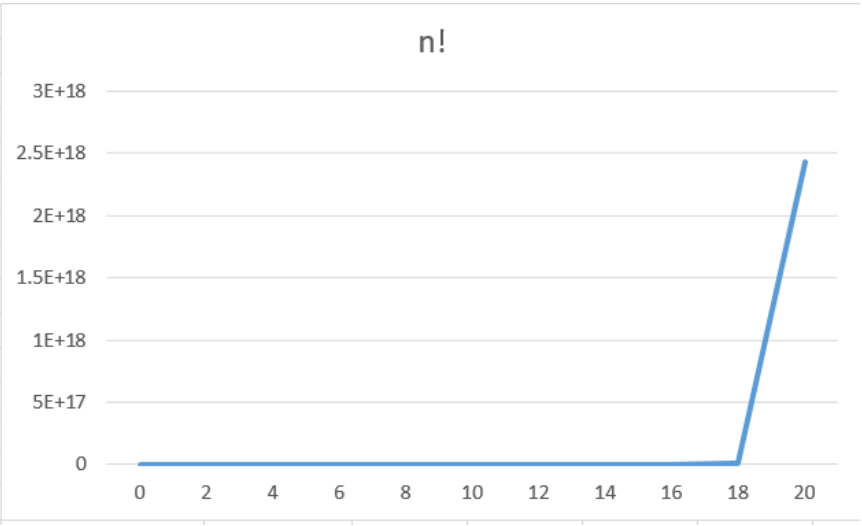
9. lnln(n)

n	lnln
0	-1.#IND00
10	0.834032
20	1.097189
30	1.224128
40	1.305323
50	1.364055
60	1.409607
70	1.446565
80	1.477511
90	1.504035
100	1.52718



10. n!

n	n!
0	1
2	2
4	24
6	720
8	40320
10	3628800
12	479001600
14	87178291200
16	2.09228E+13
18	6.40237E+15
20	2.4329E+18



Aim – Experiment on finding the running time of an algorithm.

Insertion sort

It works similar to the sorting of playing cards in hands. It is assumed that the first card is already sorted in the card game, and then we select an unsorted card. If the selected unsorted card is greater than the first card, it will be placed at the right side; otherwise, it will be placed at the left side. Similarly, all unsorted cards are taken and put in their exact place.

Selection sort

It first finds the smallest value among the unsorted elements of the array is selected in every pass and inserted to its appropriate position into the array. In this algorithm, the array is divided into two parts, first is sorted part, and another one is the unsorted part. Initially, the sorted part of the array is empty, and unsorted part is the given array. Sorted part is placed at the left, while the unsorted part is placed at the right. In selection sort, the first smallest element is selected from the unsorted array and placed at the first position. After that second smallest element is selected and placed in the second position. The process continues until the array is entirely sorted.

Experiment1b.c > main()

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

void swap(int*x, int *y){
    int temp = *x;
    *x = *y;
    *y = temp;
}

void printArray(int *arr, int n){
    for(int i = 0; i < n; i++){
        printf("%d\n", arr[i]);
    }
}

//Selection Sort
void selectionSortt(int *arr, int start, int size){
    for(int i = start; i < size; i++){
        int minIdx = i;
        for (int j = i; j < size; j++){
            if(arr[j] < arr[minIdx])
            {
                minIdx = j;
            }
            swap(&arr[i], &arr[minIdx]);
        }

    }
}

//Isertion Sort
void insertionSortt(int *arr, int start, int size){
    for(int i = start+1; i < size; i++){
        int temp = arr[i];
        int j = i - 1;
```

Experiment1b.c > main()

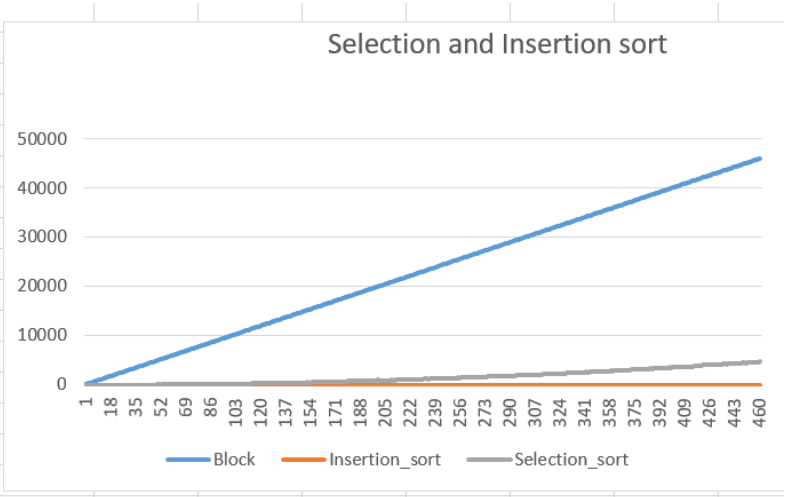
```c
        while(j >= 0 && temp <= arr[j]){
            arr[j+1] = arr[j];
            j = j - 1;
        }
        arr[j+1] = temp;
    }
}

int main(){
    printf("\n");
    char *randNumders = "randNumber.txt";
    char *selectionSort = "selction.txt";
    char *insertionSort = "insertion.txt";

    FILE *randnumfptr = fopen(randNumders, "w");
    FILE *sortednumfptr;

    int n = 100000;
    int a[n];

    int num, startTime, endTime, rangeTill;

    //Created random array
    printf("Generating random array");
    for(int i = 0; i < n; i++){
        num = rand();
        fprintf(randnumfptr, "%d\n", num);
    }
    fclose(randnumfptr);

    //Insertion sort
    randnumfptr = fopen(randNumders, "r");
    printf("\nReading random array");
    for(int i = 0; i < n; i++){
        fscanf(randnumfptr, "%d" , &a[i]);
    }
    fclose(randnumfptr);
```

Experiment1b.c > main()

```c
    fclose(randnumfptr);
    printf("\nDone\n");
    rangeTill = 100;

    printf("Insertionsort\n");
    sortednumfptr = fopen(insertionSort, "w");
    printf("Sorted numbers stored in (%s)",insertionSort);

    while(rangeTill <= n){
        startTime = clock();
        insertionSortt(a, 0,rangeTill);
        endTime = clock();
        fprintf(sortednumfptr, "%d\n",endTime-startTime);
        rangeTill += 100;
    }
    fclose(sortednumfptr);
    printf("\nInsertion sort done");


    //Selection Sort
    randnumfptr = fopen(randNumders, "r");
    printf("\nReading random array");
    for(int i = 0; i < n; i++){
        fscanf(randnumfptr, "%d", &a[i]);
    }
    fclose(randnumfptr);

    printf("\nDone\n");
    rangeTill = 100;

    printf("Selectionsort\n");
    sortednumfptr = fopen(selectionSort, "w");
    printf("Sorted numbers stored in (%s)",selectionSort);

    while(rangeTill <= n){
        startTime = clock();
        selectionSortt(a , 0 , rangeTill);
```

```c
C Experiment1b.c > main()
110             selectionSortt(a , 0 , rangeTill);
111             endTime = clock();
112             fprintf(sortednumfptr, "%d\n", endTime-startTime);
113             rangeTill += 100;
114         }
115         fclose(sortednumfptr);
116         printf("\nSelection sort done");
117         return 0;
118     }
```

Block	Insertion_sort	Selection_sort
100	0	0
200	0	0
300	0	0
400	0	0
500	0	1
600	0	1
700	1	0
800	0	3
900	0	2
1000	0	2
1100	0	5
1200	0	3
1300	0	4
1400	1	6
1500	0	6
1600	0	6
1700	0	8
1800	1	7
1900	0	10
2000	0	9
2100	0	12
2200	1	12
2300	0	11
2400	0	2
2500	0	16



Conclusion: In this experiment, we implemented, calculated & analyzed the runtime of various functions by plotting their outputs in range 0-100 with increment of 10. Also, we calculated & analyzed the time consumed by insertion & selection sorting algorithms