

<b>Name</b>	Nishant Dinesh Satere
<b>UID no.</b>	202271009
<b>Experiment No.</b>	1

<b>AIM:</b>	Experiment on finding the running time of algorithms
-------------	--

### Program 1

<b>PROBLEM STATEMENT :</b>	To find the running time of various algorithms
----------------------------	--

<b>PROGRAM:</b>	<pre> C Experiment1.c X C: &gt; Users &gt; Admin &gt; Desktop &gt; SPIT &gt; Sem4 &gt; Practicals &gt; DDA &gt; C Experiment1.c &gt; function3() 1  #include&lt;stdio.h&gt; 2  #include&lt;math.h&gt; 3 4  //To implement the various functions e.g. linear, non-linear, quadratic, exponential etc. 5 6  //(3/2)^n 7  void function1() 8  { 9      for(int n = 0; n &lt;= 100; n = n+10) 10     { 11         printf("%d : %f\n", n , pow(1.5,n)); 12     } 13 } 14 15 //n^3 16 void function2() 17 { 18     for(int n = 0; n &lt;= 100; n=n+10) 19     { 20         printf("%d : %d\n",n,n*n*n); 21     } 22 } 23 24 //2^(2^n) 25 void function3() 26 { 27     for(int n = 0; n &lt;= 100; n = n+10) 28     { 29         printf("%u : %f\n",n, 2*(pow(2,n))); 30     } 31 } 32 33 //log2(n) 34 void function4() 35 { 36     for(int n = 0; n &lt;= 100; n=n+10) 37     { </pre>
-----------------	--

--	--

C Experiment1.c X

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

```
38     printf("%d : %f\n", n ,log2(n));
39     }
40 }
41
42 //2^(log2(n))
43 void function5()
44 {
45     for(int n = 0; n <= 100; n= n+10)
46     {
47         printf("%d : %f\n",n,pow(2,(log2(n))));
48     }
49 }
50
51 //n
52 void function6()
53 {
54     for(int n = 0; n <= 100; n = n+10)
55     {
56         printf("%d : %d\n",n,n);
57     }
58 }
59
60 //2^n
61 void function7()
62 {
63     for (int n = 0; n <= 100; n = n+10)
64     {
65         printf("%d : %f\n",n,pow(2,n));
66     }
67 }
68
69 //nlog2(n)
70 void function8()
71 {
72     for(int n = 0; n <= 100; n = n+10)
73     {
74
```

C Experiment1.c X

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

```
75         printf("%d : %f\n",n,n*log2(n));
76     }
77 }
78
79 //2^((2^n)+1)
80 void function9()
81 {
82     for(int n =0; n <= 100; n = n+10)
83     {
84         printf("%d : %f\n",n,pow(2,pow(2,n+1)));
85     }
86 }
87
88 //lnln(n)
89 void function10()
90 {
91     for(int n = 0; n <= 100; n = n+10)
92     {
93         printf("%d : %f\n",n,log(log(n)));
94     }
95 }
96
97 //n!
98 void function11()
99 {
100     for(int n = 0; n <= 20; n = n+2)
101     {
102         double fact = 1;
103         for(int i = 1; i <= n; i++)
104         {
105             fact = fact*i;
106         }
107         printf("%d : %f\n",n,fact);
108     }
109 }
110
111 int main()
```

```

112 {
113     printf("Press 1 : Function1 (3/2)*n\n");
114     printf("Press 2 : Function2 n^3\n");
115     printf("Press 3 : Function3 2^(2^n)\n");
116     printf("Press 4 : Function4 log2(n)\n");
117     printf("Press 5 : Function5 2^(log2(n))\n");
118     printf("Press 6 : Function6 n\n");
119     printf("Press 7 : Function7 2^n\n");
120     printf("Press 8 : Function8 nlog2(n)\n");
121     printf("Press 9 : Function9 2^((2^n)+1)\n");
122     printf("Press 10 : Function10 lnln(n)\n");
123     printf("Press 11 : Function11 n!\n");
124     printf("Press 12 : Exit\n");
125
126     int option;
127     while (option != 12)
128     {
129         printf("Enter your choice : ");
130         scanf("%d" ,&option);
131         switch (option)
132         {
133             case 1:
134                 printf("Function1 (3/2)*n : \n");
135                 function1();
136                 break;
137
138             case 2:
139                 printf("Function2 n^3 : \n");
140                 function2();
141                 break;
142
143             case 3:
144                 printf("Function3 2^(2^n) : \n");
145                 function3();
146                 break;
147
148             case 4:

```

Experiment1.c X

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

```
149     printf("Function4 log2(n) : \n");
150     function4();
151     break;
152
153     case 5:
154     printf("Function5 2^(log2(n)) : \n");
155     function5();
156     break;
157
158     case 6:
159     printf("Function6 n : \n");
160     function6();
161     break;
162
163     case 7:
164     printf("Function7 2^n : \n");
165     function7();
166     break;
167
168     case 8:
169     printf("Function8 nlog2(n) : \n");
170     function8();
171     break;
172
173     case 9:
174     printf("Function9 2^((2^n)+1) : \n");
175     function9();
176     break;
177
178     case 10:
179     printf("Function10 lnln(n) : \n");
180     function10();
181     break;
182
183     case 11:
184     printf("Function11 n! : \n");
185     function11();
186
187     break;
188 }
189 }
```

## OUTPUT:

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

```
PS C:\Users\Admin\Desktop\SPIT\sem4\Practicals\DDA> gcc Experiment1.c
```

```
PS C:\Users\Admin\Desktop\SPIT\sem4\Practicals\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 : 219902325552.000000
50 : 2251799813685248.000000
60 : 2305843009213694000.000000
70 : 2361183241434822600000.000000
80 : 2417851639229258300000000.000000
90 : 2475880078570760500000000000.000000
100 : 253530120045645880000000000000.000000
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 : 126765060022822940000000000000.000000

Enter your choice : 8

Function8  $\text{nlog}_2(n)$  :

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

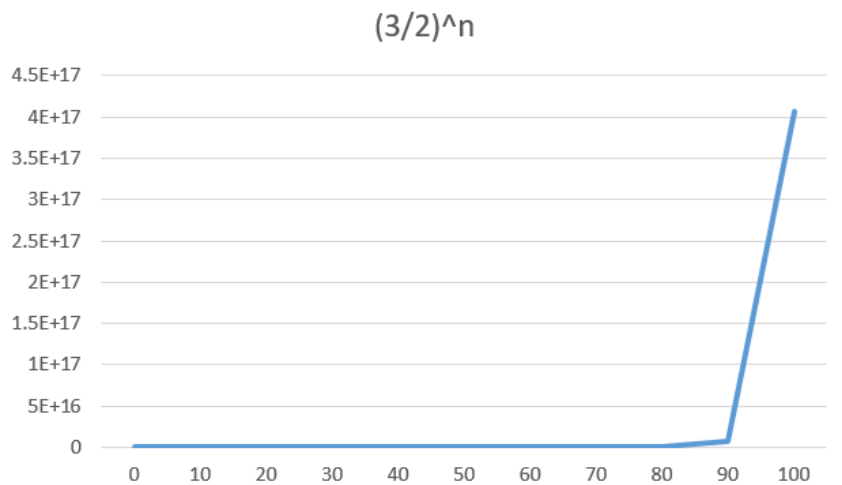
Enter your choice : 9

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  $\ln(n)$  :
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\Practicals\DDA> |
```

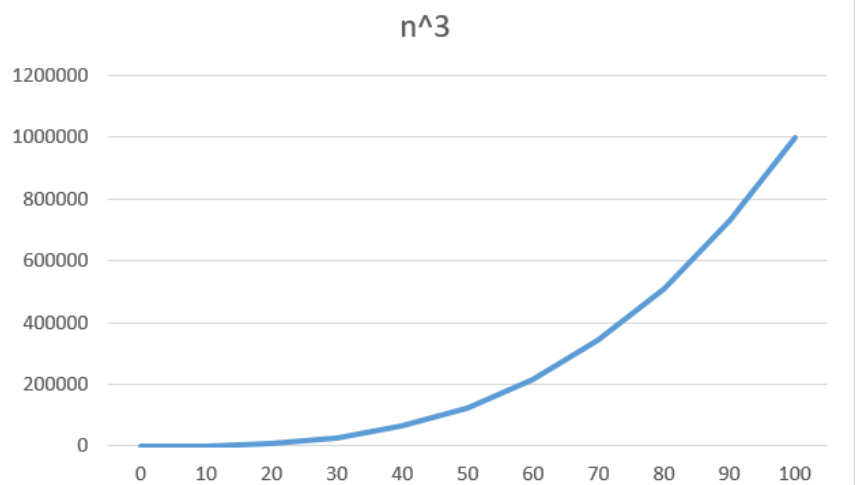
1.  $(3/2)^n$

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



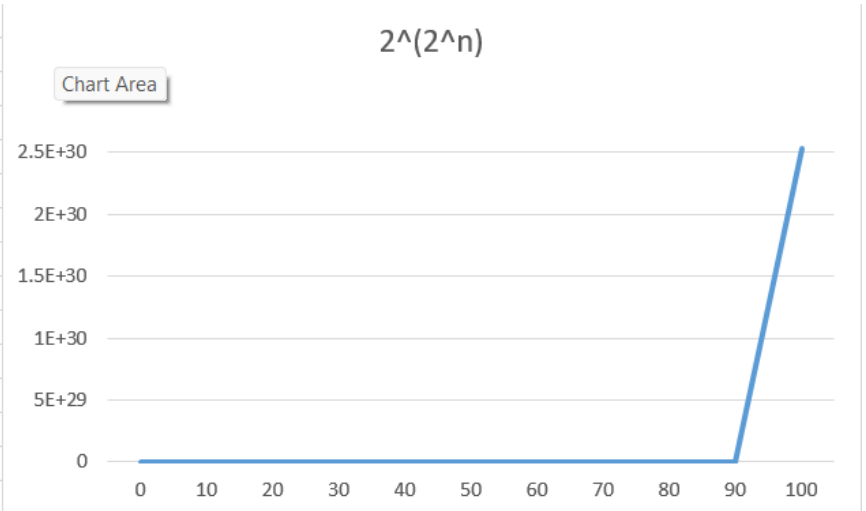
2.  $n^3$

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



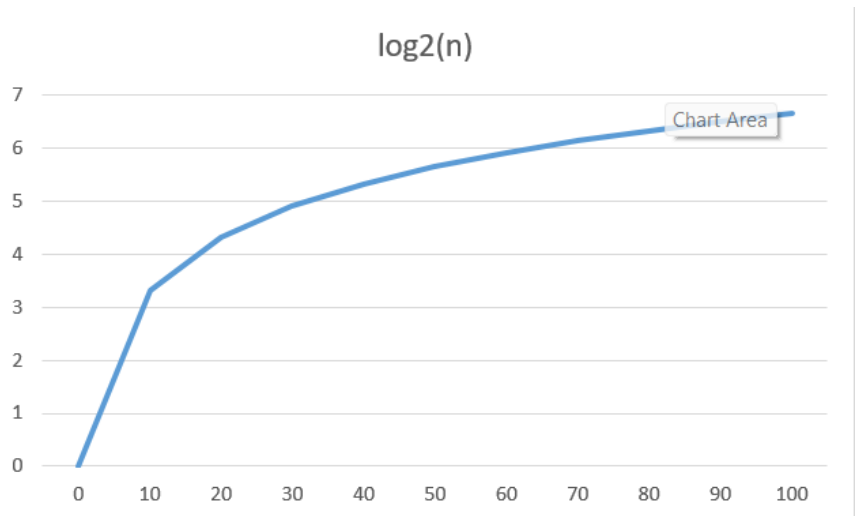
### 3. $2^{(2^n)}$

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



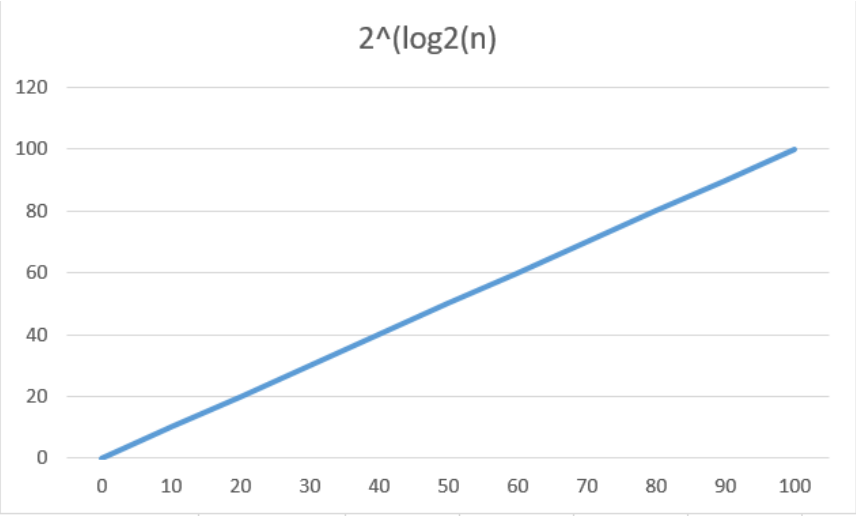
### 4. $\log_2(n)$

n	$\log_2(n)$
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



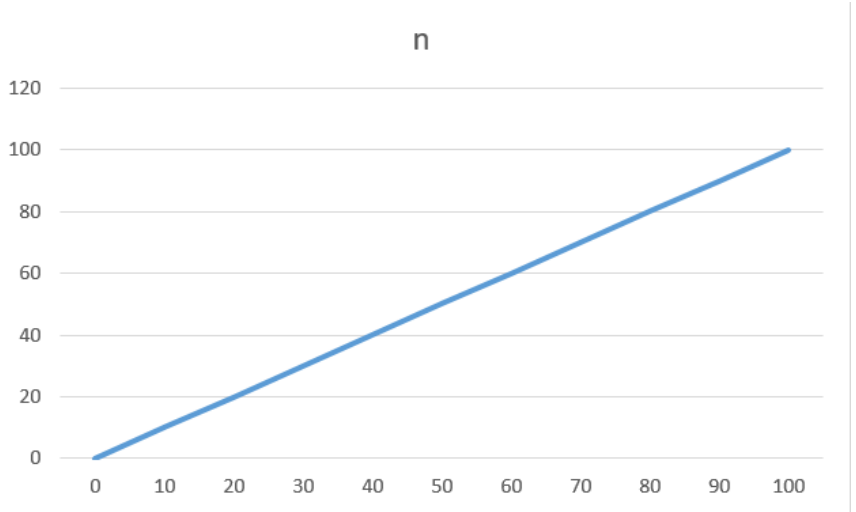
5.  $2^{\log_2(n)}$

n	$2^{\log_2(n)}$
0	0
10	10
20	20
30	30
40	40
50	50
60	60
70	70
80	80
90	90
100	100



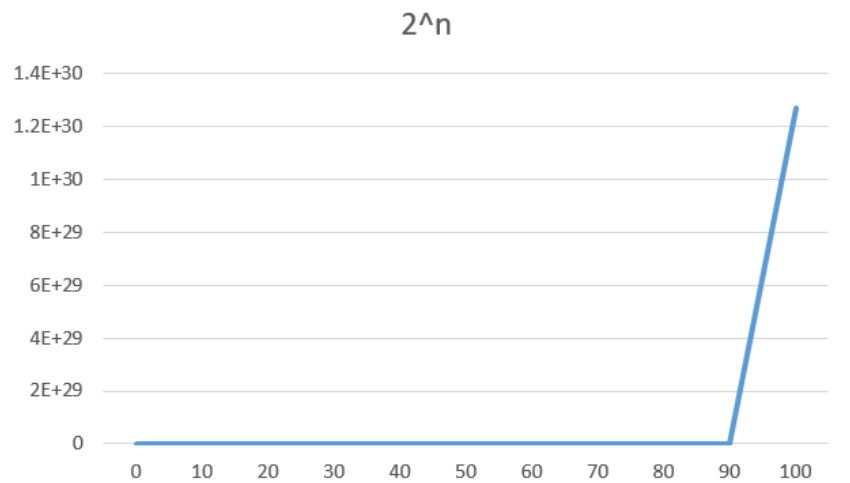
6.  $n$

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



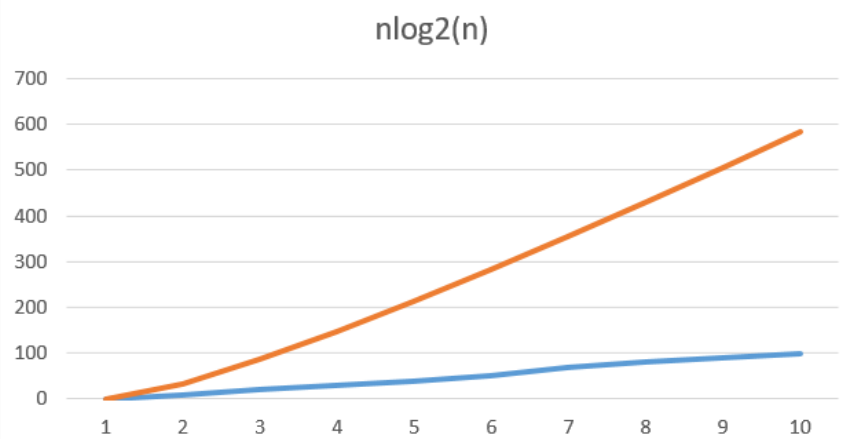
7.  $2^n$

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

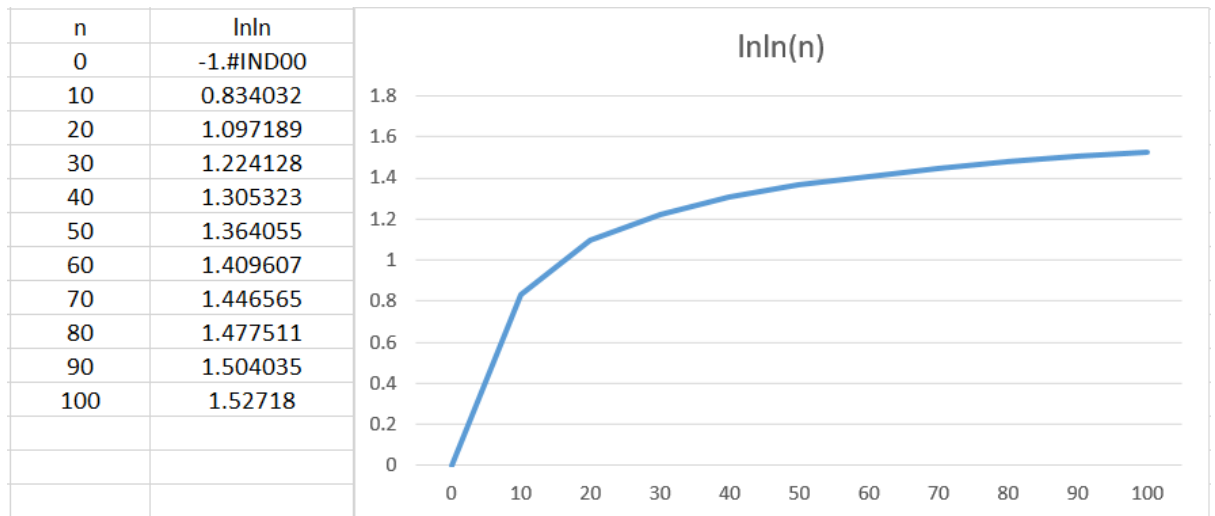


8.  $n \log_2(n)$

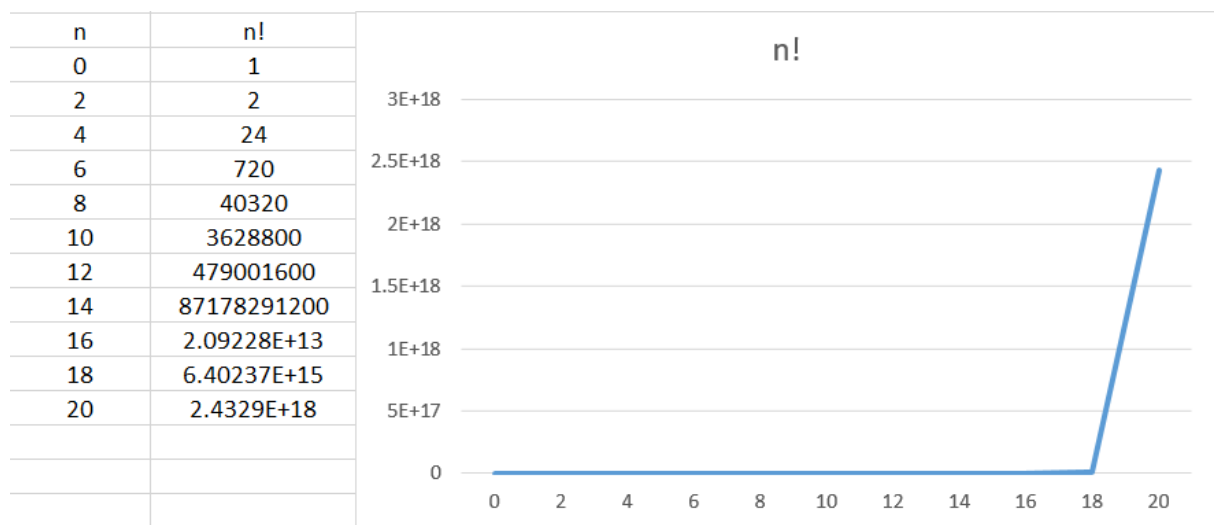
n	$n \log_2(n)$
0	nan
10	33.219281
20	86.438562
30	147.206718
40	212.877124
50	282.192809
70	354.413436
80	429.049811
90	505.754248
100	584.266779



### 10. $\ln \ln(n)$



### 11. $n!$



## Program 2

**PROBLEM  
STATEMENT :**

Implement two sorting algorithms namely Insertion and Selection sort methods. Compare these algorithms based on time and space complexity.

**ALGORITHM/  
THEORY:****Insertion Sort**

```
procedure insertionSort( A : array of items )
  int holePosition
  int valueToInsert
  for i = 1 to length(A) inclusive do:
    /* select value to be inserted */
    valueToInsert = A[i]
    holePosition = i
    /*locate hole position for the element to be inserted */
    while holePosition > 0 and A[holePosition-1] > valueToInsert do:
      A[holePosition] = A[holePosition-1]
      holePosition = holePosition - 1
    end while

    /* insert the number at hole position */
    A[holePosition] = valueToInsert
  end for
end procedure
```

**Selection Sort**

```
procedure selection sort
  list : array of items
  n : size of list
  for i = 1 to n - 1
    /* set current element as minimum*/
    min = i
    /* check the element to be minimum */
    for j = i+1 to n
      if list[j] < list[min] then
        min = j;
      end if
    end for
    /* swap the minimum element with the current element*/
    if indexMin != i then
```



	swap list[min] and list[i] end if end for end procedure
<b>PROGRAM:</b>	<pre> C Experiment1b.c &gt; main() 1  #include&lt;stdio.h&gt; 2  #include&lt;math.h&gt; 3  #include&lt;stdlib.h&gt; 4  #include&lt;time.h&gt; 5 6  void swap(int*x, int *y){ 7      int temp = *x; 8      *x = *y; 9      *y = temp; 10 } 11 12 void printArray(int *arr, int n){ 13     for(int i = 0; i &lt; n; i++){ 14         printf("%d\n", arr[i]); 15     } 16 } 17 18 //Selection Sort 19 void selectionSortt(int *arr, int start, int size){ 20     for(int i = start; i &lt; size; i++){ 21         int minIdx = i; 22         for (int j = i; j &lt; size; j++){ 23             if(arr[j] &lt; arr[minIdx]) 24             { 25                 minIdx = j; 26             } 27             swap(&amp;arr[i], &amp;arr[minIdx]); 28         } 29     } 30 } 31 32 33 //Isertion Sort 34 void insertionSortt(int *arr, int start, int size){ 35     for(int i = start+1; i &lt; size; i++){ 36         int temp = arr[i]; 37         int j = i - 1; </pre>

```

C Experiment1b.c > main()
38     while(j >= 0 && temp <= arr[j]){
39         arr[j+1] = arr[j];
40         j = j - 1;
41     }
42     arr[j+1] = temp;
43 }
44 }
45
46 int main(){
47     printf("\n");
48     char *randNumders = "randNumber.txt";
49     char *selectionSort = "selction.txt";
50     char *insertionSort = "insertion.txt";
51
52     FILE *randnumfptr = fopen(randNumders, "w");
53     FILE *sortednumfptr;
54
55     int n = 100000;
56     int a[n];
57
58     int num, startTime, endTime, rangeTill;
59
60     //Created random array
61     printf("Generating random array");
62     for(int i = 0; i < n; i++){
63         num = rand();
64         fprintf(randnumfptr, "%d\n", num);
65     }
66     fclose(randnumfptr);
67
68     //Insertion sort
69     randnumfptr = fopen(randNumders, "r");
70     printf("\nReading random array");
71     for(int i = 0; i < n; i++){
72         fscanf(randnumfptr, "%d" , &a[i]);
73     }
74     fclose(randnumfptr);

```

C Experiment1b.c > main()

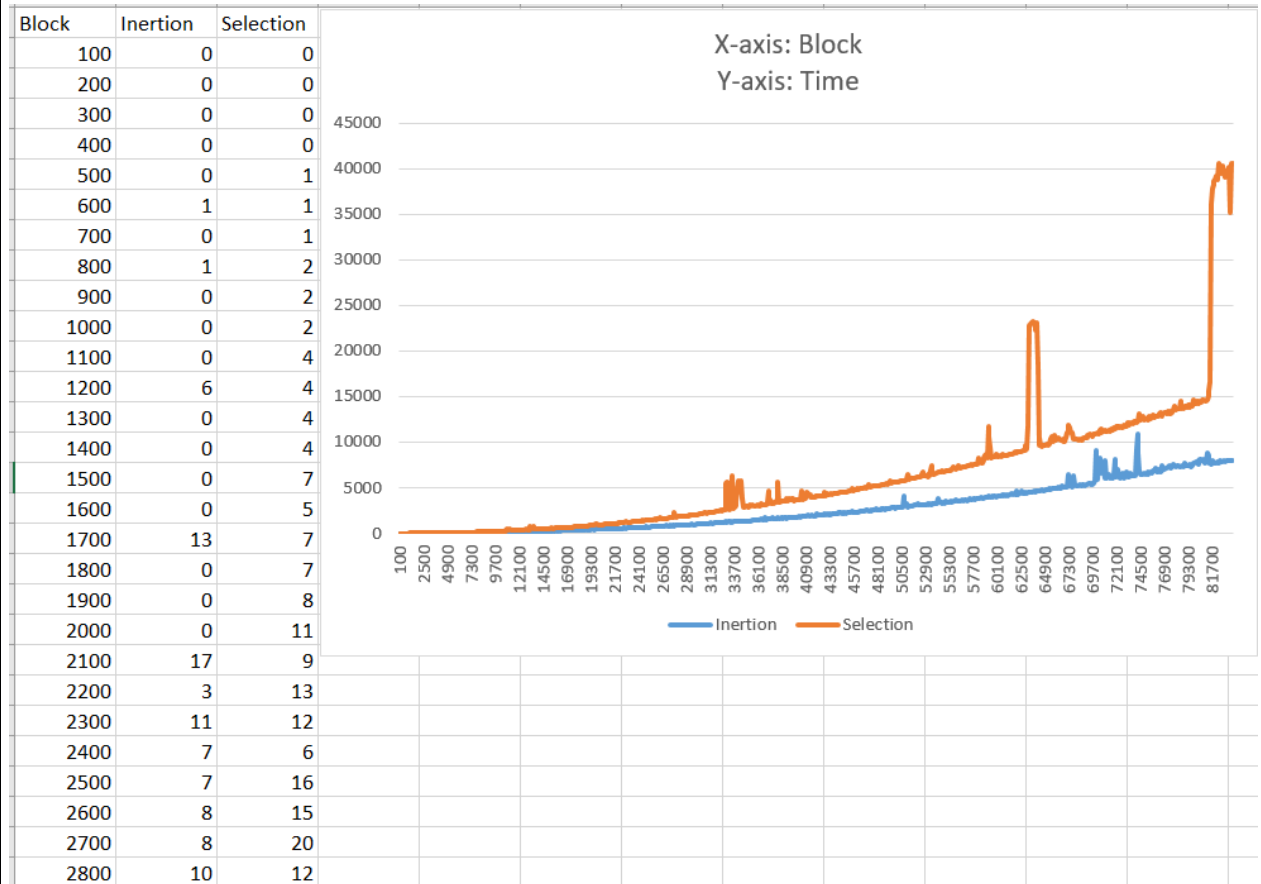
```
74     fclose(randnumfptr);
75     printf("\nDone\n");
76     rangeTill = 100;
77
78     printf("Insertionsort\n");
79     sortednumfptr = fopen(insertionSort, "w");
80     printf("Sorted numbers stored in (%s)",insertionSort);
81
82     while(rangeTill <= n){
83         startTime = clock();
84         insertionSortt(a, 0,rangeTill);
85         endTime = clock();
86         fprintf(sortednumfptr, "%d\n",endTime-startTime);
87         rangeTill += 100;
88     }
89     fclose(sortednumfptr);
90     printf("\nInsertion sort done");
91
92
93     //Selection Sort
94     randnumfptr = fopen(randNumbers, "r");
95     printf("\nReading random array");
96     for(int i = 0; i < n; i++){
97         fscanf(randnumfptr, "%d", &a[i]);
98     }
99     fclose(randnumfptr);
100
101     printf("\nDone\n");
102     rangeTill = 100;
103
104     printf("Selectionsort\n");
105     sortednumfptr = fopen(selectionSort, "w");
106     printf("Sorted numbers stored in (%s)",selectionSort);
107
108     while(rangeTill <= n){
109         startTime = clock();
110         selectionSortt(a , 0 , rangeTill);
```

```

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 }

```

## GRAPH:



## 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