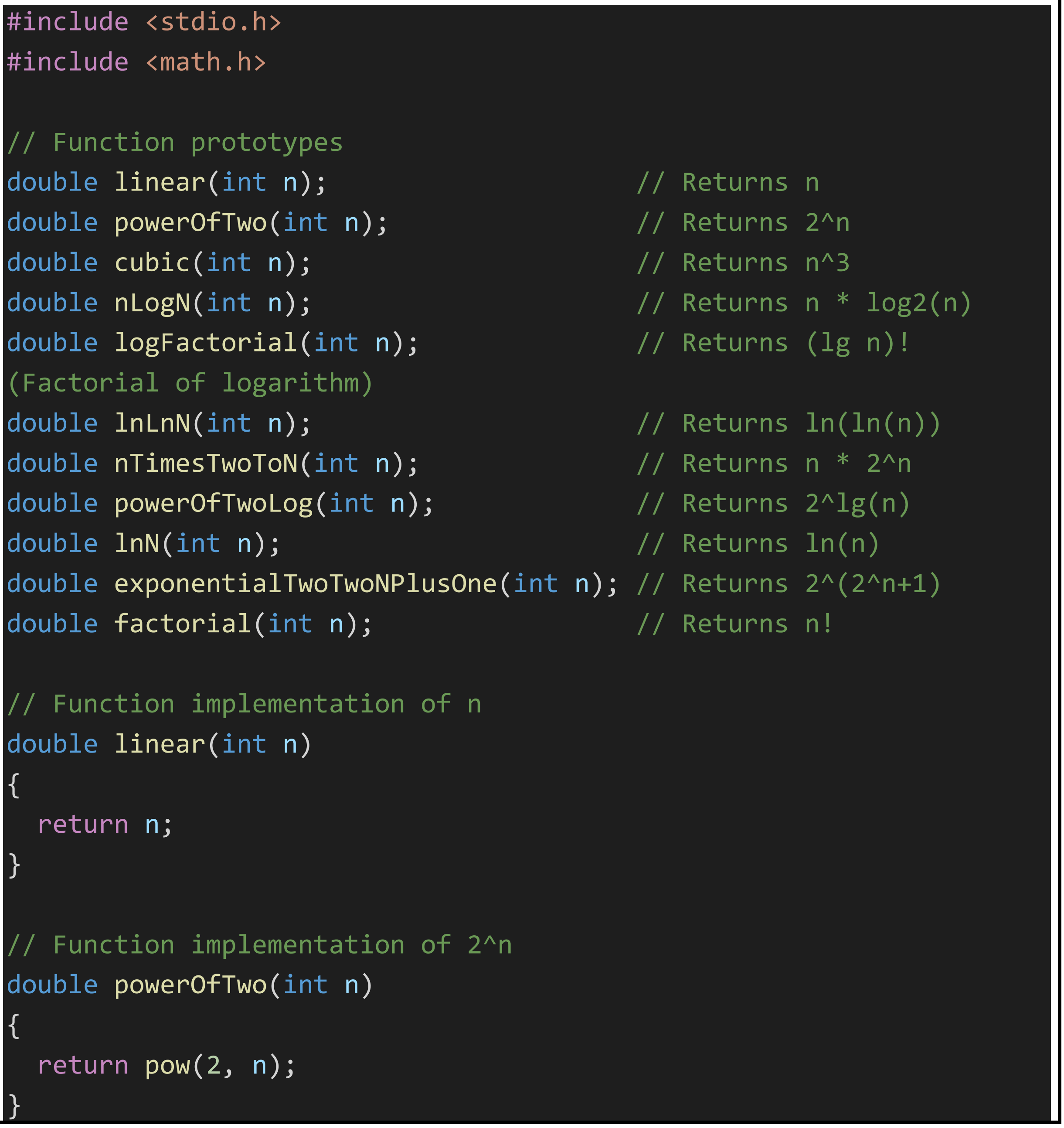
BHARATIYA VIDYA BHAVAN'S
SARDAR PATEL INSTITUTE OF TECHNOLOGY
(Empowered Autonomous Institute Affiliated to Mumbai University)
Department Of Computer Engineering

Name	Manish Shashikant Jadhav
UID	2023301005
Subject	Design and Analysis of Algorithms.
Experiment No.	1.A
Aim	To implement the various functions e.g. linear, non-linear, quadratic, exponential etc.

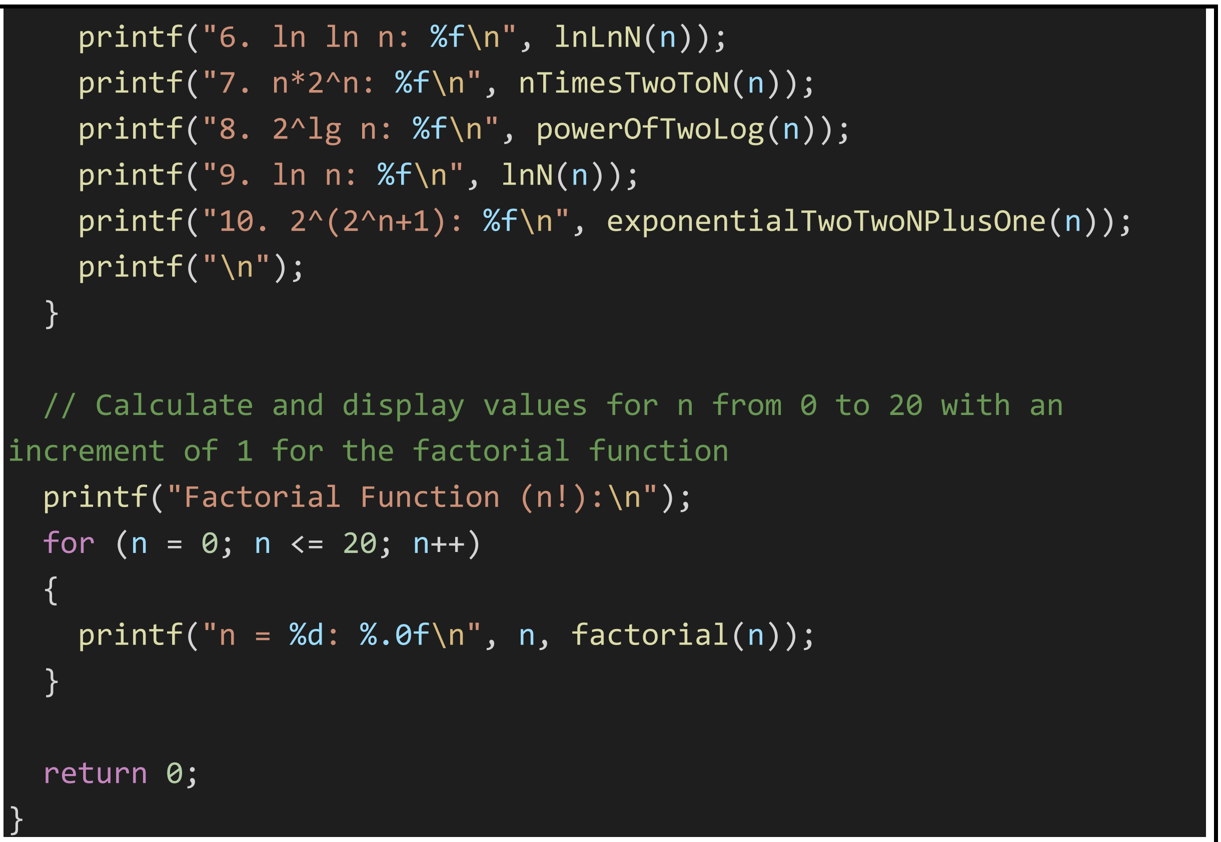
Code



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

// Function prototypes
double linear(int n);                    // Returns n
double powerOfTwo(int n);                // Returns 2^n
double cubic(int n);                     // Returns n^3
double nLogN(int n);                     // Returns n * log2(n)
double logFactorial(int n);              // Returns (lg n)!
(Factorial of logarithm)
double lnLnN(int n);                     // Returns ln(ln(n))
double nTimesTwoToN(int n);              // Returns n * 2^n
double powerOfTwoLog(int n);             // Returns 2^lg(n)
double lnN(int n);                       // Returns ln(n)
double exponentialTwoTwoNPlusOne(int n); // Returns 2^(2^n+1)
double factorial(int n);                 // Returns n!

// Function implementation of n
double linear(int n)
{
  return n;
}

// Function implementation of 2^n
double powerOfTwo(int n)
{
  return pow(2, n);
}
```

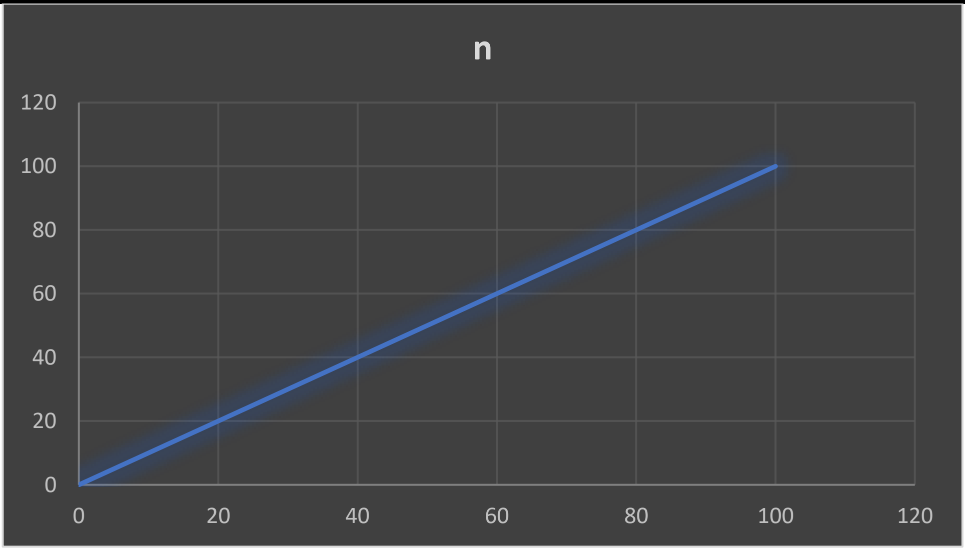
```
// Function implementation of n^3
double cubic(int n)
{
  return pow(n, 3);
}

// Function implementation of n * log2(n)
double nLogN(int n)
{
  return n * log2(n);
}

// Function implementation of (lg n)! (Factorial of logarithm)
double logFactorial(int n)
{
  return tgamma(log2(n) + 1); // Using tgamma for gamma function (factorial )
}

// Function implementation of ln(ln(n))
double lnLnN(int n)
{
  return log(log(n));
}

// Function implementation of n * 2^n
double nTimesTwoToN(int n)
{
  return n * pow(2, n);
}

// Function implementation of 2^lg(n)
double powerOfTwoLog(int n)
{
  return pow(2, log2(n));
}
```

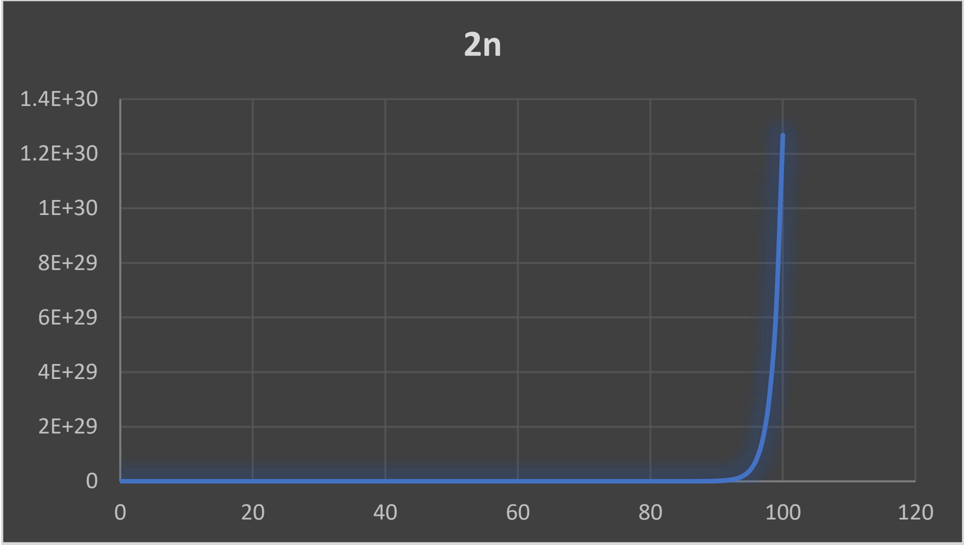
```

// Function implementation of ln(n)
double lnN(int n)
{
  return log(n);
}

// Function implementation of 2^(2^n+1)
double exponentialTwoTwoNPlusOne(int n)
{
  return pow(2, pow(2, n) + 1);
}

// Function implementation of n!
double factorial(int n)
{
  if (n == 0 || n == 1)
    return 1;
  else
    return n * factorial(n - 1);
}

int main()
{
  int n;

  // Calculate and display values for n from 0 to 100 with an increment of 1
  for (n = 0; n <= 100; n++)
  {
    printf("n = %d:\n", n);
    printf("1. n: %d\n", (int)linear(n));
    printf("2. 2^n: %f\n", powerOfTwo(n));
    printf("3. n^3: %f\n", cubic(n));
    printf("4. n lg n: %f\n", nLogN(n));
    printf("5. (lg n)! (Factorial of logarithm): %f\n", logFactorial(n));
```

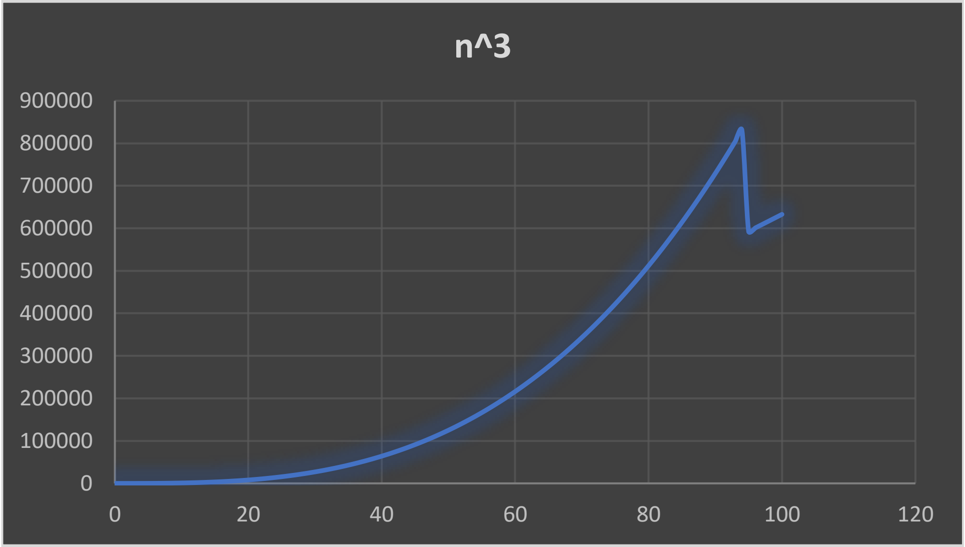


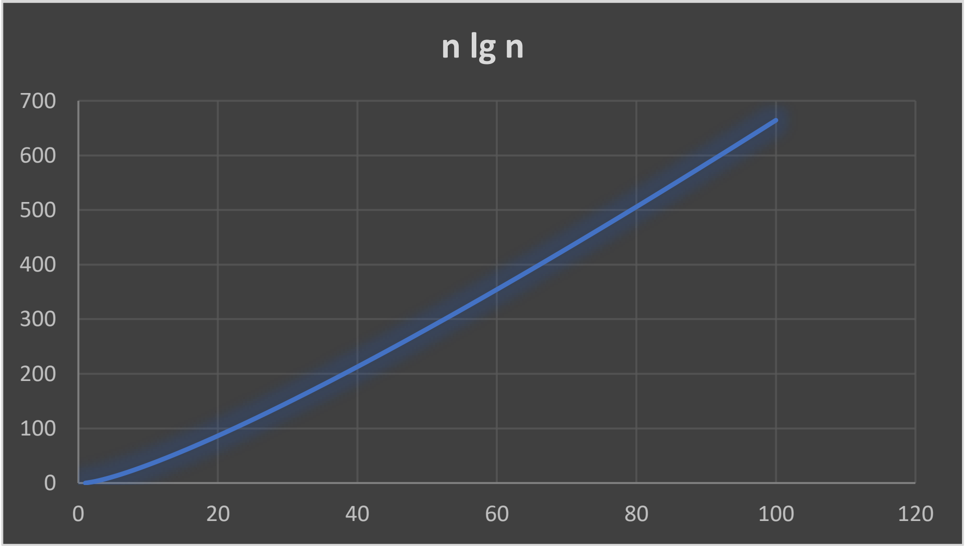
```
    printf("6. ln ln n: %f\n", lnLnN(n));
    printf("7. n*2^n: %f\n", nTimesTwoToN(n));
    printf("8. 2^lg n: %f\n", powerOfTwoLog(n));
    printf("9. ln n: %f\n", lnN(n));
    printf("10. 2^(2^n+1): %f\n", exponentialTwoTwoNPlusOne(n));
    printf("\n");
  }

  // Calculate and display values for n from 0 to 20 with an increment of 1 for the factorial function
  printf("Factorial Function (n!):\n");
  for (n = 0; n <= 20; n++)
  {
    printf("n = %d: %.0f\n", n, factorial(n));
  }

  return 0;
}
```

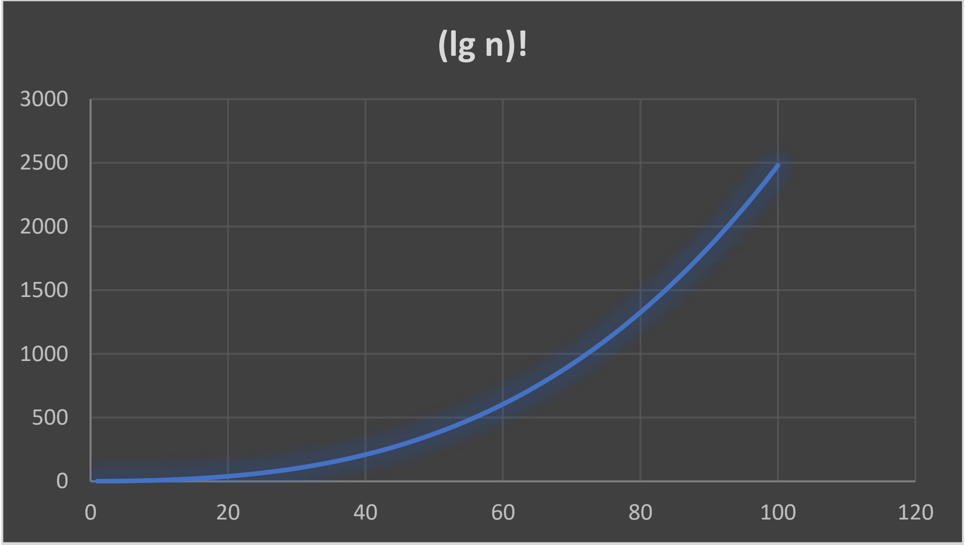
Graph

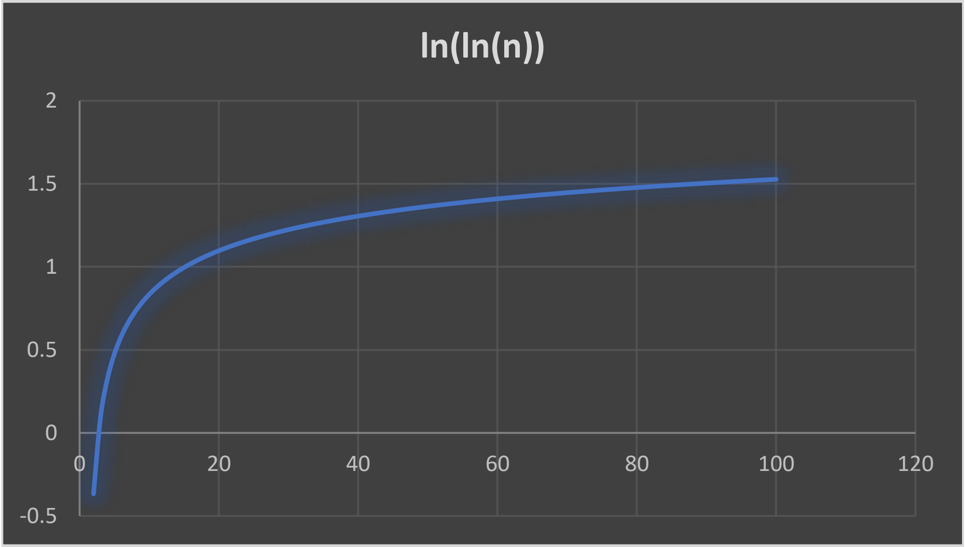


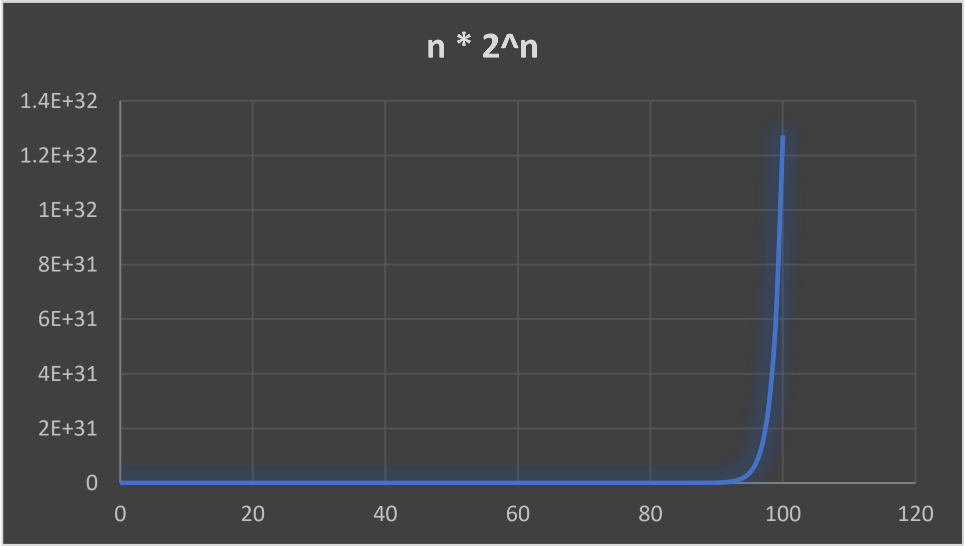

2n
1.4E+30
1.2E+30
1E+30
8E+29
6E+29
4E+29
2E+29
0
0
20
40
60
80
100
120

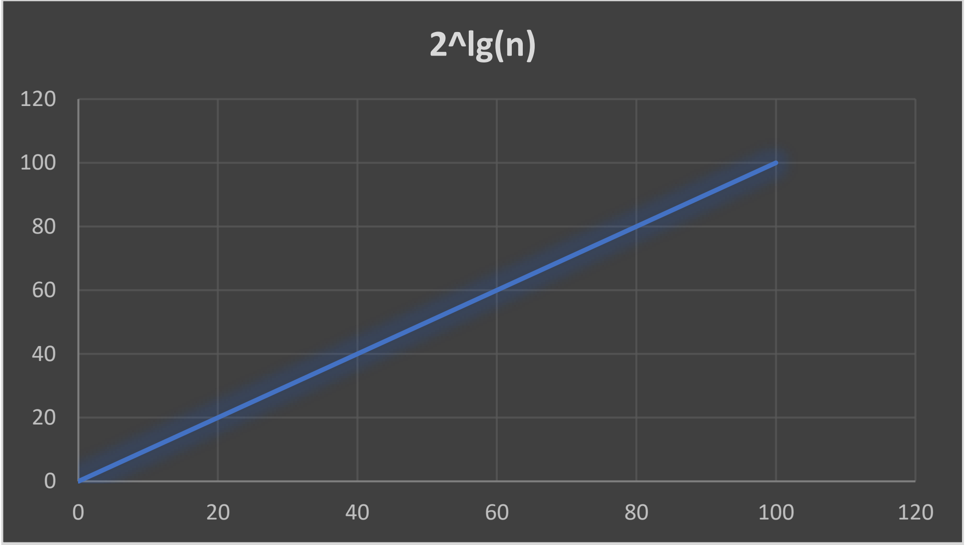

n^3
900000
800000
700000
600000
500000
400000
300000
200000
100000
0
0
20
40
60
80
100
120

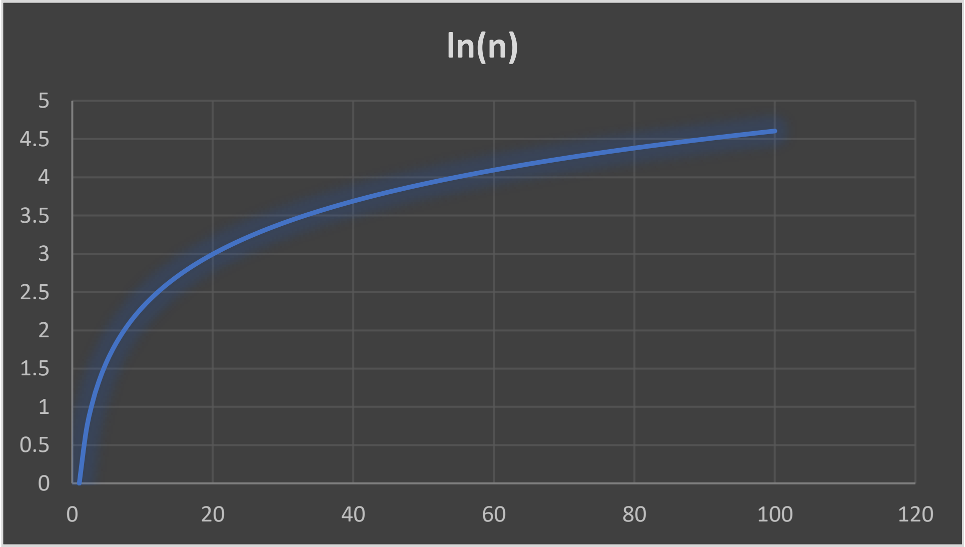

n lg n
700
600
500
400
300
200
100
0
0
20
40
60
80
100
120

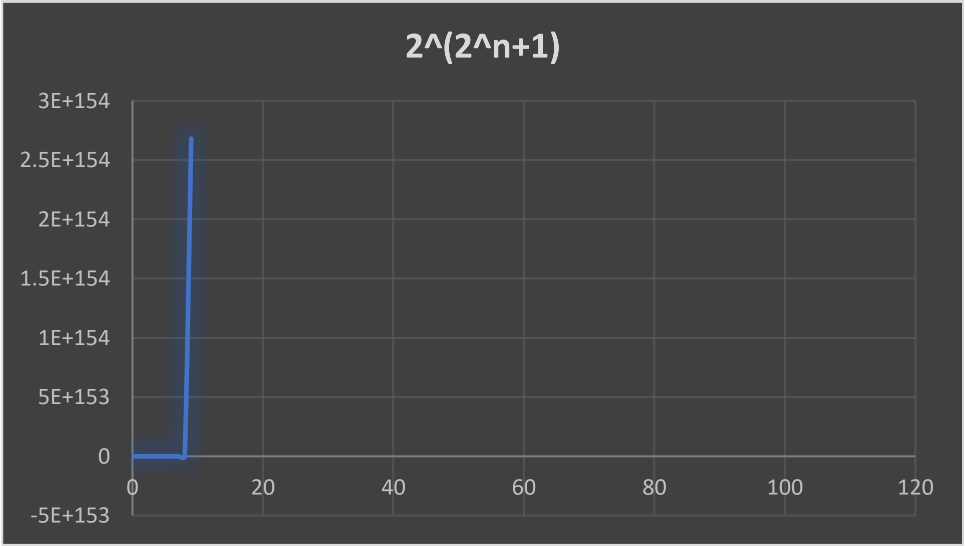

(lg n)!
3000
2500
2000
1500
1000
500
0
0
20
40
60
80
100
120

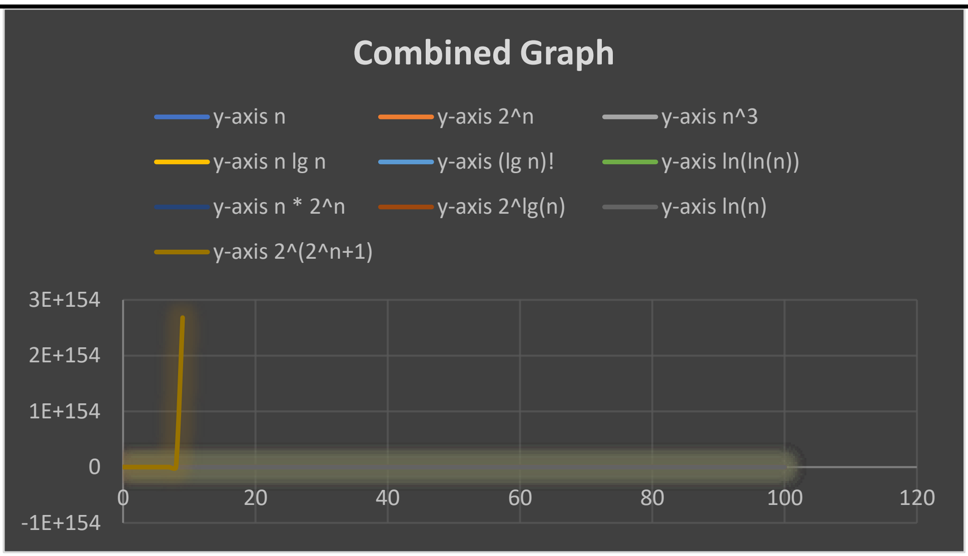

ln(ln(n))
2
1.5
1
0.5
0
-0.5
0
20
40
60
80
100
120


n * 2^n
1.4E+32
1.2E+32
1E+32
8E+31
6E+31
4E+31
2E+31
0
0
20
40
60
80
100
120

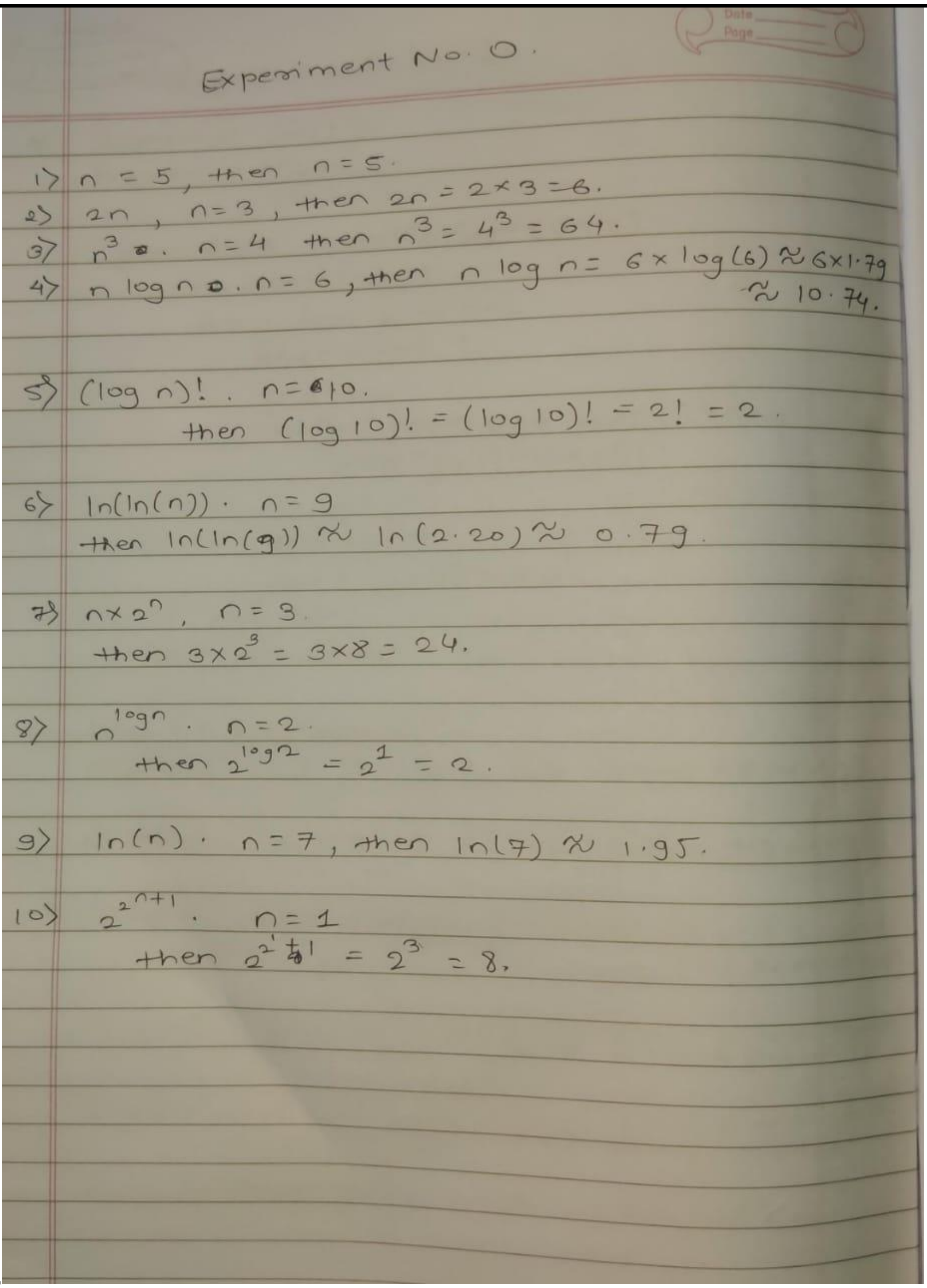

2^lg(n)
120
100
80
60
40
20
0
0
20
40
60
80
100
120

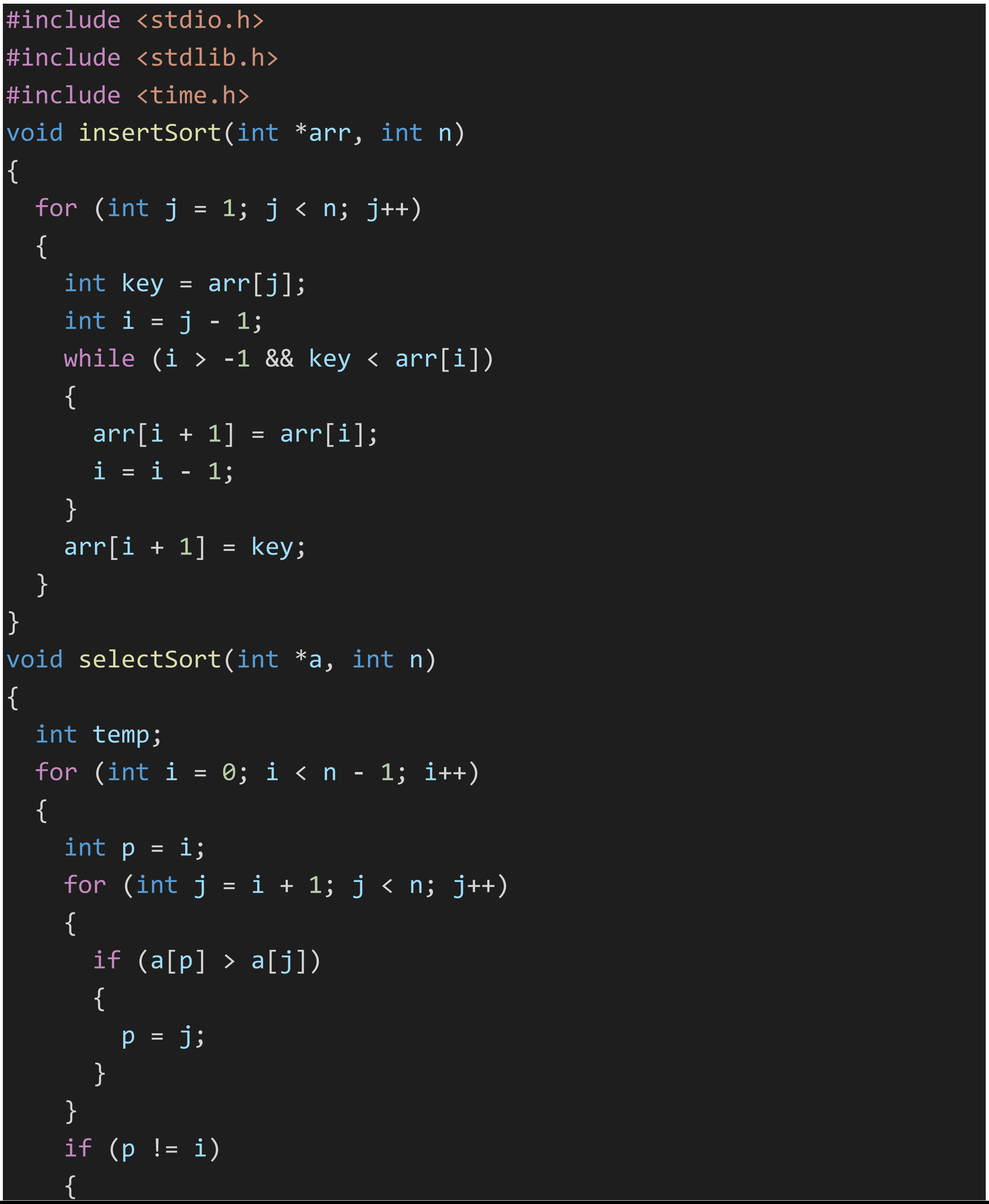

ln(n)
5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
0
20
40
60
80
100
120


2^(2^n+1)
3E+154
2.5E+154
2E+154
1.5E+154
1E+154
5E+153
0
-5E+153
0
20
40
60
80
100
120

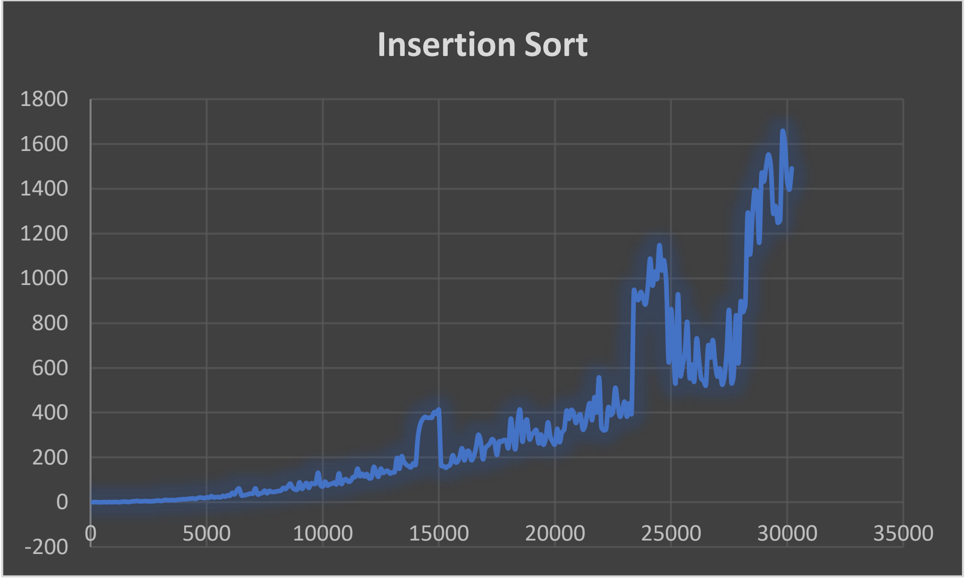
Combined Graph
y-axis n
y-axis 2^n
y-axis n^3
y-axis n lg n
y-axis (lg n)!
y-axis ln(ln(n))
y-axis n * 2^n
y-axis 2^lg(n)
y-axis ln(n)
y-axis 2^(2^n+1)
3E+154
2E+154
1E+154
0
-1E+154
0
20
40
60
80
100
120

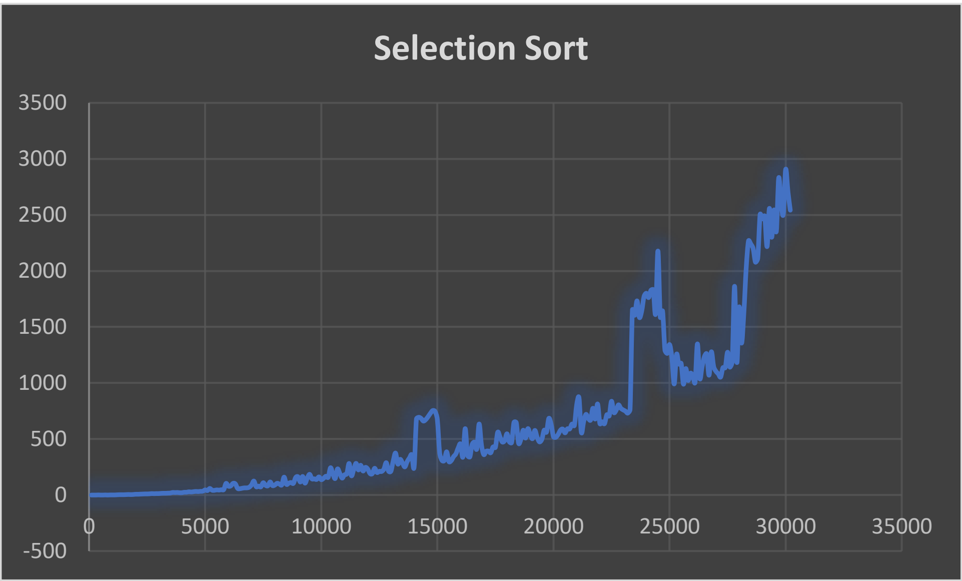
Solved Problem:	 Date Page Experiment No. 0. 1> $n = 5$, then $n = 5$. 2> $2n$, $n = 3$, then $2n = 2 \times 3 = 6$. 3> n^3. $n = 4$ then $n^3 = 4^3 = 64$. 4> $n \log n$. $n = 6$, then $n \log n = 6 \times \log(6) \approx 6 \times 1.79$ ≈ 10.74. 5> $(\log n)!$. $n = 10$. then $(\log 10)! = (\log 10)! = 2! = 2$. 6> $\ln(\ln(n))$. $n = 9$ then $\ln(\ln(9)) \approx \ln(2.20) \approx 0.79$. 7> $n \times 2^n$, $n = 3$. then $3 \times 2^3 = 3 \times 8 = 24$. 8> $n^{\log n}$. $n = 2$. then $2^{\log 2} = 2^1 = 2$. 9> $\ln(n)$. $n = 7$, then $\ln(7) \approx 1.95$. 10> 2^{2^n+1}. $n = 1$ then $2^{2^1+1} = 2^3 = 8$.
Conclusion	Hence, by completing this experiment I came to know about to implement the various functions e.g. linear, non-linear, quadratic, exponential etc.

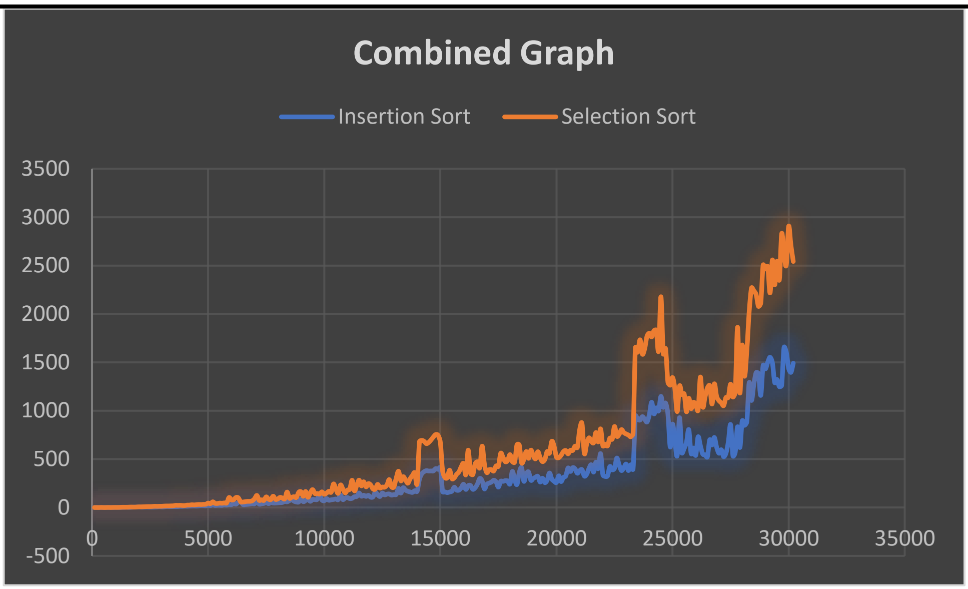
Experiment No.	1.B
Aim	Experiment on finding the running time of an algorithm(Insertion Sort and Selection Sort).
Code	

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
void insertSort(int *arr, int n)
{
  for (int j = 1; j < n; j++)
  {
    int key = arr[j];
    int i = j - 1;
    while (i > -1 && key < arr[i])
    {
      arr[i + 1] = arr[i];
      i = i - 1;
    }
    arr[i + 1] = key;
  }
}
void selectSort(int *a, int n)
{
  int temp;
  for (int i = 0; i < n - 1; i++)
  {
    int p = i;
    for (int j = i + 1; j < n; j++)
    {
      if (a[p] > a[j])
      {
        p = j;
      }
    }
    if (p != i)
    {
      temp = a[i];
      a[i] = a[p];
      a[p] = temp;
    }
  }
}
void main()
{
  srand(time(NULL));
  FILE *fileptr;
  FILE *fileptr1;
  int n = 100000;
  fileptr = fopen("inp.txt", "w");
  int arr[n];
  for (int j = 0; j < n; j++)
  {
    arr[j] = rand() % 100000 + 1;
    fprintf(fileptr, "%d\n", arr[j]);
  }
  fclose(fileptr);

  fileptr1 = fopen("time.csv", "w");
  printf("Block Size\tInsertion Sort\tSelection sort\n");

  fileptr = fopen("inp.txt", "r");
  for (int p = 99; p < n; p = p + 100)
  {
    int array[p + 1];
    int array1[p + 1];
    for (int j = 0; j < p; j++)
    {
      array[j] = arr[j];
      fscanf(fileptr, "%1d", &array1[j]);
    }
    clock_t begin = clock();
    insertSort(array, p + 1);
    clock_t mid = clock();
    selectSort(array1, p + 1);
    clock_t end = clock();
    double time_spent = (double)(mid - begin);
    double time_spent1 = (double)(end - mid);
    printf("%d\t%lf\t%lf\n", p + 1, time_spent, time_spent1);
    fprintf(fileptr1, "%d,%lf,%lf\n", p + 1, time_spent,
time_spent1);
  }
  printf("\n");
}
```

Graph:



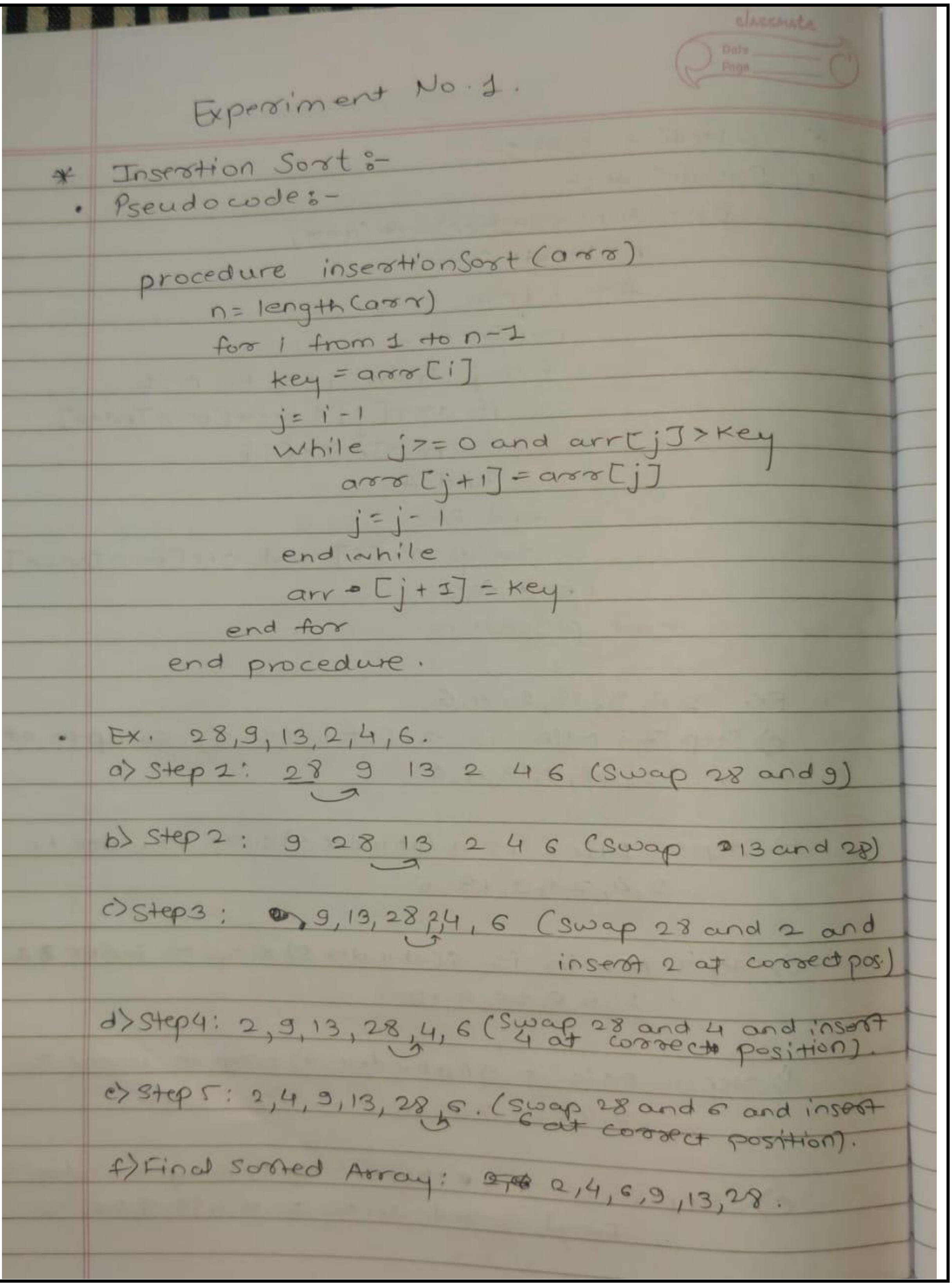


Combined Graph
Insertion Sort
Selection Sort
3500
3000
2500
2000
1500
1000
500
0
-500
0
5000
10000
15000
20000
25000
30000
35000

BHARATIYA VIDYA BHAVAN'S
SARDAR PATEL INSTITUTE OF TECHNOLOGY
(Empowered Autonomous Institute Affiliated to Mumbai University)
Department Of Computer Engineering

Insertion Sort Pseudocode and example:



Experiment No. 1.

* Insertion Sort :-

- Pseudocode:-

```
procedure insertionSort (arr)
    n= length(arr)
    for i from 1 to n-1
        key = arr[i]
        j= i-1
        while j>=0 and arr[j]>key
            arr [j+1] = arr[j]
            j=j-1
        end while
        arr [j+1] = key.
    end for
end procedure.
```

- Ex. 28,9,13,2,4,6.

a) Step 1: 28 9 13 2 4 6 (Swap 28 and 9)

b) Step 2 : 9 28 13 2 4 6 (Swap 13 and 28)

c) Step3 : 9,13,28,2,4, 6 (Swap 28 and 2 and insert 2 at correct pos.)

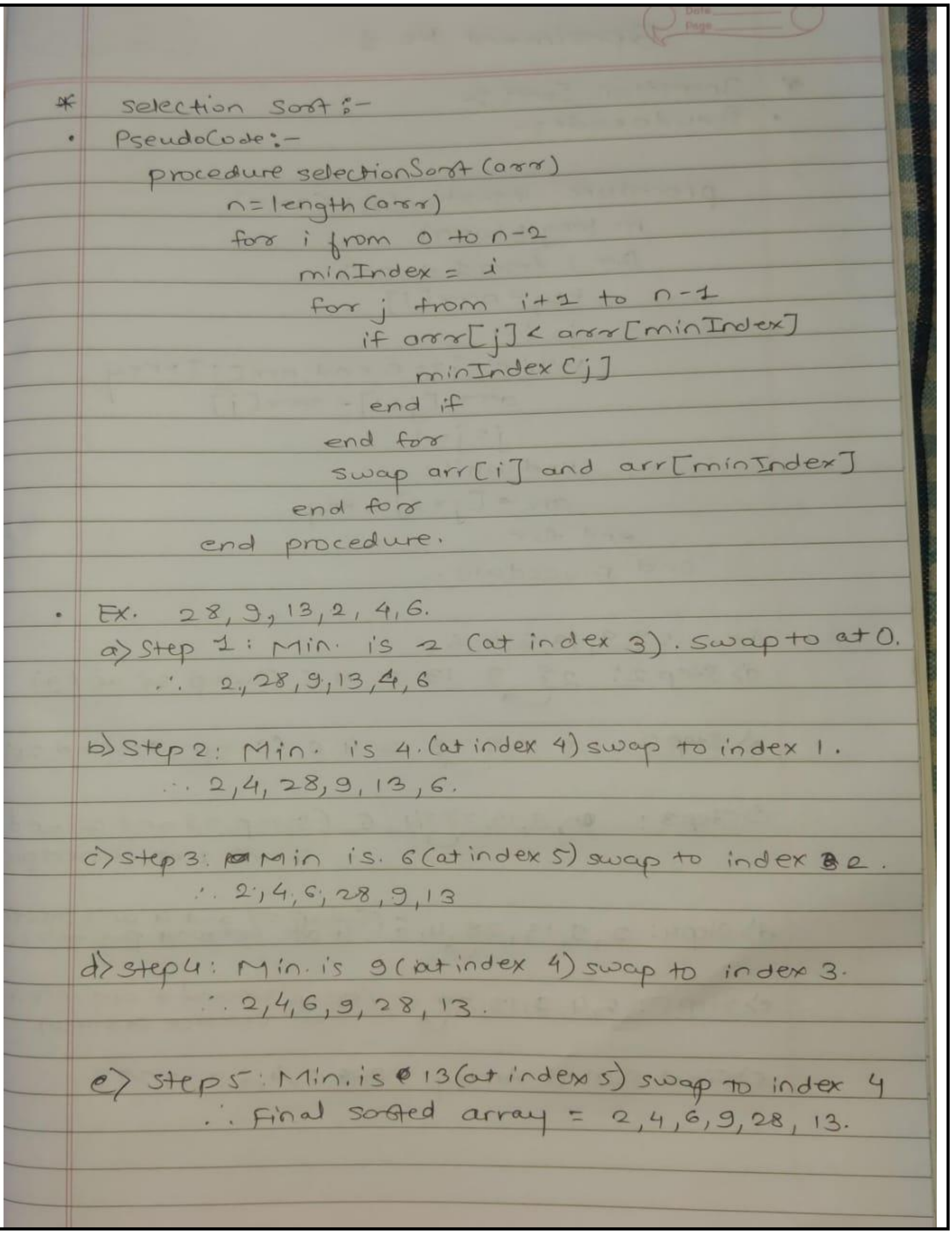
d) Step4: 2,9,13,28,4,6 (Swap 28 and 4 and insert 4 at correct position).

e) Step 5: 2,4,9,13,28,6. (Swap 28 and 6 and insert 6 at correct position).

f) Final sorted Array: 2,4,6,9,13,28.

Selection Sort

Pseudocode and example:



* Selection Sort :-

- PseudoCode :-

```
procedure selectionSort (arr)
    n= length (arr)
    for i from 0 to n-2
        minIndex = i
        for j from i+1 to n-1
            if arr[j] < arr[minIndex]
                minIndex [j]
            end if
        end for
        swap arr[i] and arr[minIndex]
    end for
end procedure.
```

- Ex. 28, 9, 13, 2, 4, 6.

a) Step 1: Min. is 2 (at index 3). Swap to at 0.
∴ 2, 28, 9, 13, 4, 6

b) Step 2: Min. is 4. (at index 4) swap to index 1.
∴ 2, 4, 28, 9, 13, 6.

c) Step 3: Min is. 6 (at index 5) swap to index 2.
∴ 2, 4, 6, 28, 9, 13

d) step 4: Min. is 9 (at index 4) swap to index 3.
∴ 2, 4, 6, 9, 28, 13.

e) step 5: Min. is 13 (at index 5) swap to index 4
∴ Final sorted array = 2, 4, 6, 9, 28, 13.

Conclusion	Hence, by completing this experiment I came to know about finding the running time of an algorithm(Insertion Sort and Selection Sort).