**Write everything including the code in your script and submit via google form.**

**Topic: Time complexity, Searching and Sorting**

1. **Time Complexity and Sorting (40 Marks)**
   a. **Sort** the following functions in descending order of their growth. For example- $n^2<n^3$, etc **using Selection sort. Show the simulation in details.**
   $2^n$, log n, $n^2\log(n^2)$, log(log n), $n^2$, n, $\sqrt{n}$ , n!, $n^3$, $n^{3/2}$, nlogn, $e^{n+1}$, $n^2$ log n
   **(8 marks)**

   b. Prove **T(n)= 2T(n-1)+n** can be represented in $\theta(n2^n)$using telescoping method **(4 Marks**)

   c. **Prove** the following (you can use any formal induction/other theoretical method, "^" means power here):
      i. What is the time complexity recurrence relation for Fibonacci numbers? Explain it with the subproblems and size of sub-problems. Show the time complexity derivation using the telescoping method. (4 marks)

      ii. Show that 2n^3 + 5n^2 + 6n + 18 is in Theta (n^3). ( 2 marks)
      iii. 5 + 2cos(n) = Theta (1) (2 marks)
      iv. Show that (3/2)n^2 +2n-3 = Ω (n^2) (2 marks)
      v. Show that n^3+5n is not O(n^2) (2 marks).

      vi. $T(n) = 2T(n/2) + n^3 = O(n^3)$ (2 marks)
      vii. $T(n) = T(n/4) + T(5n/8) + n = O(n)$ (2 $marks$)

   d. **Let, the time complexity of each of the following code snippets be T(n).** Find out a tight bound for T(n) in Big-Theta (θ) notation. **3*4 marks**

      1. **p=0**
         **for (i=1, i<=n; i++)**
           **while(p<i):**
              **p=p+i**

      2. **for (i=n, i>=1; i=i/3)**
           **p=2**
           **while(p<n)**
              **print("hello")**
              **p=p*p**

```
3. p =0
  for i=1;i<n;i=i*2
        p++
  for j=1;j<p;j=j*2
        print("hello")
```

**2. Searching (10 marks)**

You are given two arrays: Array1 and Array2.

Array1 will be sorted. For each element v in Array2, you need to **write a pseudocode** that will print the number of elements in Arr1 that is **greater than or equal to v and the element must be a prime number**. For example: if I give you two arrays of size 5 and 4

5 4 [size of two arrays]

Arr1 = 1 3 5 7 9

Arr2 = 6 4 8

The output should be: 1 2 0

There are two parts in the question. First you need to know how many numbers are there in Array1 which are greater than 6. Here, we find there are two numbers (7,9). Then we need to find whether these numbers are prime or not. The searching portion must not take more than **O(log n)** time.

| Sample input | Sample output |
|---|---|
| 5 5<br>3 7 10 12 17<br>6 1 14 8 9 | 2 3 1 1 1 |