.

Techniques to calculate Time Complexity:

Once we are able to write the runtime in terms of the size of the input (n), we can find the time complexity. For example:

$T(n) = n2 \rightarrow O(n^2)$

$T(n) = logn \rightarrow O(logn)$

Here are some tricks to calculate complexities:

- Drop the constants:

Anything you might think is O(kn) (where k is a constant) is O(n) as well. This is considered a better representation of the time complexity since the k term would not affect the complexity much for a higher value of n.

- Drop the non-dominant terms: :

Anything you represent as O(n2+n) can be written as O(n2). Similar to when non-dominant terms are ignored for a higher value of n.

- Consider all variables which are provided as input:

O (mn) and O (mnq) might exist for some cases.

In most cases, we try to represent the runtime in terms of the inputs which can even be more than one in number. For example,

The time taken to paint a park of dimension m * n → O (kmn) → O (mn)

Time Complexity – Competitive Practice Sheet:

Question1: Fine the time complexity of the func1 function in the program shown in the snippet below:

```c
#include<stdio.h>

void func1(int array[], int length)
{
    int sum=0;
    int product =1;
    for (int i = 0; i <length; i++)
    {
        sum+=array[i];
    }

    for (int i = 0; i < length; i++)
    {
        product*=array[i];
    }
}

int main()
{
    int arr[] = {3,4,66};
    func1(arr,3);
    return 0;
}
```

Question 2: Find the time complexity of the func function in the program from program2.c as follows:

```
void func(int n)
{
    int sum=0;
    int product =1;
    for (int i = 0; i <n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            printf("%d , %d\n", i,j);
        }
    }
}
```

Question 3: Consider the recursive algorithm below, where the random(int n) spends one unit of time to return a random integer where the probability of each integer coming as random is evenly distributed within the range [0,n]. If the average processing time is T(n), what is the value of T(6)?

```
int function(int n)
{
    int i = 0;
    if (n <= 0)
    {
        return 0;
    }
    else
```

```
    {

        i = random(n - 1);

        printf("this\n");

        return function(i) + function(n - 1 - i);

    }

}
```

Question 4: Which of the following are equivalent to O(N) and why?

O(N + P), where P < N/9

0(9N-k)

O(N + 8log N)

O(N + M2)

Question 5: The following simple code sums the values of all the nodes in a balanced binary search tree (don't worry about what it is, we'll learn them later). What is its runtime?

```
int sum(Node node)

{

    if (node == NULL)

    {

        return 0;

    }

    return sum(node.left) + node.value + sum(node.right);

}
```

Question 6: Find the complexity of the following code which tests whether a given number is prime or not?

```
int isPrime(int n)

{

    if (n == 1)

    {

        return 0;

    }

    for (int i = 2; i * i < n; i++)

    {

        if (n % i == 0)

        {

            return 0;

        }

    }

    return 1;

}
```

Question 7: What is the time complexity of the following snippet of code?

```
int isPrime(int n)

{

    for (int i = 2; i * i < 10000; i++)

    {

        if (n % i == 0)

        {

            return 0;

        }
```

```
    }


    return 1;

}

isPrime();
```