Date. _____
Page No. _____

ASSIGNMENT-1

Q.1. What do you understand by Asymptotic relations. Define different Asymptotic relation with example.

Ans: Asympatotic notations are a set of mathematical tools used to describe the behavior of function as their input sizes approach Infinity. They are often used to analyse the time and space complexity of algorithms.

There are 3 main types of asymptotic relation:

1. Big O notation (O):

This notation provides an upper bound on the growth rate of a function. It represents the worst-case running time of an algorithm, which is the maximum amount of time it could take to complete. for example we say that an algo has time complexity of O(n), we mean that the algorithm's running time grows at most, linearly with the size of its input.

ex:-
```
int sum = 0;
for (int i = 1; i <= n; i++)
{

sum += i;
}                    // The time complexity is O(n)
```

2. Omega notation (Ω):

This notation provides a lower bound on the growth rate of a function it represents the best-case running time of an algorithm, which is the minimum amount of time it could take to complete for example if we say that an algorithm has a time complexity of $\Omega(n)$, we mean that the algorithm's running time grows at least linearly with the size of its input.

Date.
Page No.

3. Theta notation (Θ):

This notation provides both an upper and a lower bound on the growth rate of a function it represent the avg-case running time of an algorithm, which is the expected amount of time it would take to complete. For example, if we say that an algorithm has a time complexity of $O(n)$, we mean that the algorithm's running time grows linearly with the size of its inputs and these are no faster or slower growth rates.

Ex: def bubble sort (lst):

```
    n= len (lst)
    for i in range (n):
     for j in range (n-i-1);
     if lst [j] > lst [j+1] = lst [j+1], lst [j]
     return lst
```

Time complexity is $\Theta(n^2)$

Q.2. What should be time complexity of for(i=1 to n) {i=i*2}

Ans: The time complexity of the loop

```
for (i= 1 to n)
{
i = i*2;
}
```

can be obtained by counting the no. of iteration that loop will execute as a function of input size 'n'.

Here value of i is being doubled in each iteration, loop terminate when i becomes $\geq n$

$2^k = n \quad \therefore k = \log(n)$

Time complexity = $O(\log n)$

Date. _____
Page No. _____

Q.3. $T(n) = \{ 3T(n-1)$ if $n>0$, otherwise $1\}$

$T(n) = 3T(n-1)$ ——— (1)

$T(n) = aT(n-b) + f(n)$ — (2)

Comparing eq (i) & (ii)

$a = 3$

$b = 1$

Since $f(n) = 0$; $k = 0$

We know that, in subtract and conquer theorem,

$T(n) = O(n^k \leq^{i=0 \text{ to } n/2} a^i)$

$\therefore T(n) = O(n^0 \leq^{i=0 \text{ to } 6} a^2)$

$T(n) = O(3^{n/1})$

$T(n) = O(3^n)$ Ans:

Q.4. $T(n) = \{2T(n-1)-1$ if $n>0$, otherwise $1\}$

$T(0) = 1$

$T(1) = 2T(0) - 1 = 1$

$T(2) = 2T(1) - 1 = 1$

$T(3) = 2T(2) - 1 = 1$

$T(4) = 2T(3) - 1 = 1$

and so on

So $T(n) = 1$ for any value of n

that is it is constant

Time complexity $= O(1)$

Date.
Page No.

Q.5. What is the time complexity of:

```
int i=1, s=1;
while (s<=n)
{
i++;
s = s+ i;
printf ("#");
}
```

Time complexity = O (sqrt (n))

The loop iterates until the value of s becomes >n.

At each iteration i is incremented by 1, and s is updated to s+i,

No. of iterations required to reach $s>n$, $s + (i+1)>n$

→ $i^2 + i - 2(n-s) > 0$, solved by quadratic formula

Q6.

```
void function (int n){
{
int i, count =0;
for (i=1; i*i<=n); i++)
    count ++;
}
}
```

Time complexity of the given function is O (sqrt (n)).

The for loop iterates from i=1 to i, i * i <=n.

The loop will execute for all values of i from 1 to the largest integer is less than or equal to square root of n.

Date. _____
Page No. _____

Q.7: void function (int n) {
int i, j, k, count=0;
for (i=n/2; i<=n; i++)
for (j=1; j<=n; j=j*2)
for (k=1; k<n; k=k*2)
count++;
}

Time complexity is $O(n^2 \log(n))$
The function consist of 3 nested loop that iterate variable i, j & k. the first loop takes n/2 iterations. The second loop takes iteration over variable j from 1 to n in power of 2 which takes $\log_2(n)$ iteration, third loop iterates over variable k from 1 to n in power of 2.

Q.8: function (int n) {
if (n==1)
return;
for (i=1 to n)
{
for (j=1 to n)
{ printf ("* "); }
function (n-3); }

The function is a recursive functions that is called with argument n-3, it contains two nested loop that iterate over the variable i and j. The outer loop iterate n times and inner loops also iterates n time
∴ n * n times. At each recursive call, the value of n is decreased by 3. The function will be called a total of n/3 times recursively untill n=1.
Time complexity is $O(n^2 (n/3)^2)$.

Date. _____
Page No. _____

Q.9: void function (int n) {
for (i = 1 to n)
{ for (j = 1; j <= n ; j = j + i)
printf ("* ")
}}

This function consist of two nested loop that iterate over the variable i and j the outer loop iterate over n times and inner loop iterate n/i times $n + n/2 + n/3 + \ldots + 1$.
This is Harmonic series
$\log(n) + 0.57 + 2 + O(1/n)$
Time complexity is $O(n \log \log(n))$.

Q.10: for the function, n^k and c^n. What is the asymphoric notation b/w these function.
Assume that $k >= 1$ and $c > 1$ are constants. find out the value of c and no for which relations holds.
$n^k = O(c^n)$ as n approaches infinity
n^k is bounded above by c^n.