Question 1 :
What is the time complexity T(n) for the following code snippet?

```
a = 1
b = 1
while(b <= n)
{
a += 1
b += 1
cout<<"Hi";
}
```

Solution:

In the given code snippet a=1, b=1 and we are given a while loop.

Each of assignment statements and other operations except the loop, is of constant time complexity.

Now for the loop we have

```
While(b<=n)

{

a+=1

b+=1

cout<<"Hi";

}
```

At first b=1 and it will check the condition in the while loop if it is true. Then if it is true the control will go inside the loop and increase the value of b by 1, by the statement b+=1;

Now b=b+1=2 and again the condition in the loop will be checked. If it is true, control will go inside the loop and and increase the value of b by 1 and now b=b+1=3.

This will continue until b=n;

So the statement will be executed n times. So the time complexity of this code snippet is: O(n);

So T(n)=O(n).

For an example if n=5, the at first

b=1 ------> b=2 -----> b=3---->b=4------>b=5------>b=6 and at this point loop will get terminated.

So 5 time the control will go into the loop.

So T(n)=O(n)[5 here].

Question 2 :
Write the output for the following recursive code snippet for n = 3:

```
void fun(int n)
{
if(n > 0)
{
cout<<n:
fun(n - 1);
cout<<n;
}
}
```

Solution: For n=3, first the function fun will take input as 3. Then inside the the if statement the condition will be checked and control will go inside the if statement.

At the next line it will be printed n, i.e. 3;

Then at the next line fun(3-1)=fun(2) will be called and the control will go inside the if statement,

And print 2;

Thereafter fun(2-1)=fun(1) will be called and again the control will go inside the if statement and print 1 and fun(1-1) =fun(0) will be called and here the recursion ends.

Then it will print 1, then 2 and then 3.

So the output will be like:

3 2 1

1 2 3.