WEEK 3: RECURSION

In weeks 1 and 2, you started getting familiar with the basics of competitive programming. Now, it's time to dive into the contest situation. This week is all about recursion!

A quick recap of recursion

Recursion refers to the function calling itself during execution. It is quite similar to loops in that it is also iterative. Recursions, however, solve problems by solving the small versions of the same problem first. The most important thing to remember in a recursive function is the terminating case or the base case.

For example, consider adding all elements in an array -

```
function sum(array a, integer n):
  if n>0: return a[n-1] + sum(a,n-1)
  return 0
```

Here, until `n>0`, `sum` is called recursively. At the base case `(n>0)`, `0` is returned and added to the partial sum of the array. That is then returned and added to the next partial sum until the last recursive call is finished.

So, `sum([1,2,3,4,5],5)` will return `5+4+3+2+1+0 = 15`

Tail recursion is a special type of recursion where the last thing that the function does is call itself. Hence, the result of the function is the result of the recursive call. Tail recursions are also special as they are equivalent to loops. Any tail recursive function can be converted to a loop and vice versa.

The reason for the same is how recursion is implemented in the computer memory. Every recursive call creates an *activation record* containing all local variables and parameters. In tail recursive functions, the *activation record* is unneeded after the recursive call and can be reused.

The body of the method is wrapped in the loop and instead of passing parameters in the recursive call, we reassign the loop variables. For example, consider the method to find the GCD of two numbers,

```
function gcd(integer a, integer b):
  if b=0: return a
  return gcd(b,a%b)
```

The recursive function is called until the second variable reaches `0`. Once it does, the value stored in the first variable at the time is returned as the result of the call which is also the result of the method.
To convert the above method to a loop,

```
input(a,b)
while b>0:
  r = a%b
  a = b
  b = r
print(a)
```

The loop executes until the second variable reaches `0`. After that, the gcd of the two numbers is stored in the first variable. Hence, tail recursion and iteration are equivalent.

Logistics and grading

This week onwards, there will be a contest opened every week on Hackerrank. This week's contest consists of 5 problems - 3 easy, 2 medium - all solvable by recursion. In fact, in some cases, you may notice that recursion is the better way to solve the problem.

Scoring in the contest is based on ACM rules. You will receive points based on how many test cases you pass and the maximum points on passing all test cases. There will also be a penalty of `60 seconds` for every wrong submission. THIS INCLUDES COMPILE TIME ERRORS. So be careful.

The contest will open on Monday (01/21) at 8am and remain open till Saturday (01/26) at 5pm. However, your time will start once you open the problem statement, so do not worry about your lab timings.

The link for the contest is https://www.hackerrank.com/week-3-cse212-1718 . Sign up for it and make sure to only open the problem statement when you intend to solve it.

The grading for the course will be as follows:

- On solving `20%` of the problems, you will receive `0.5/1.0` in this week's continuous lab eval
- On solving `50%` of the problems, you will receive `1.0/1.0` in this week's continuous lab eval
- On topping the leaderboard, you will receive `1.5/1.0` in this week's continuous lab eval
- Lastly, on solving `100%` of the problems, you will be invited to edit the next week's problems!

Remember, even passing 2 test cases out of 5 will give you partial credit. So, one strategy could be to attempt all problems and solve them partially. Of course the other strategy is to solve all problems entirely.

It's time you got started, so head on over to the contest at https://www.hackerrank.com/week-3-cse212-1718 and begin coding!!

Good luck, have fun!