TIC1001 — Introduction to Computing and Programming
National University of Singapore

Lecture 2: Branching and Looping

Sum of first n odd numbers - for loop

Using a for loop, write a function sum_odd_n that computes the sum of the first n odd numbers, i.e. it returns 1 + 3 + 5 + ... + (2n - 1). You may assume that n is always a positive integer.
You are to avoid using break or continue in your function.

In [ ]:

```
int sum_odd_n(int n)
{
    int i;
    int sum = 0;
    for (i=1 ; i<=(2*n)-1 ; i+=2)
    {
        sum += i;
    }
    return sum;
}
```

Sum of first n odd numbers - while loop

Using a while loop, write a function sum_odd_n that computes the sum of the first n odd numbers, i.e. it returns 1 + 3 + 5 + ... + (2n - 1). You may assume that n is always a positive integer.
You are to avoid using break or continue in your function.

In [ ]:

```
int sum_odd_n(int n)
{
    int i=1;
    int sum = 0;

    while (i<=(2*n)-1)
    {
            sum += i;
            i = i+2;
    }
    return sum;
}
```

Sum of first n squares - for loop

Complete the following function that uses a for loop to compute the sum of the squares of the first n numbers.

In [ ]:

```
int sum_n_squares(int n)
{
    int result = 0;
    for (int counter = 0; counter <= n; ++counter)
    {
       result = result + (counter*counter);
    }
    return result;
}
```

Sum of first n squares - while loop

Complete the following function that uses a while loop to compute the sum of the squares of the first n numbers.

In [ ]:

```
int sum_n_squares(int n)
{
    int counter = 1, result = 0;
    while (counter <= n)
    {
      result = result + (counter * counter);
      counter ++;
    }
    return result;
}
```

Factorial

The factorial of n (n!) is computed as 1 x 2 x 3 ... x n.
Implement the function factorial which takes a non-negative integer n and returns n!

In [ ]:

```
int factorial(int n)
{
    int fact = 1;
    for (int i = 1; i<=n; i++)
    {
        fact = fact * i;
    }
    return fact;
}
```

This question is a code-tracing practice. Please trace the code manually to test your understanding, before trying to compile and run it.
What is displayed after executing the following statements?

In [*]:

```
double d;
for (d = 0; d < 10.0; d += 0.1) ;
printf("%f\n", d);
```

Ans: 10.1

Because of floating point inaccuracy, adding 0.1 multiple times does not equal to exactly 10.0 but 9.99...

-END-