Week 2 - Practice

Exercise 1

Write a pseudocode to compute the following sums

a) $S = 1 + 2 + 3 + . . . + n$

b) $S = 1^2 + 2^2 + . . . + (n-1)^2 + n^2$

Exercise 1

a) $S = 1 + 2 + 3 + \ldots + n$

There are two ways of calculating S. Either you can use the formula $S = n(n+1)/2$ or we just do as we did in class.

For the first one, $S = n(n+1)/2$, it is very easy as follows:

```
Input n
    S: = n(n+1)/2        //here we compute S directly
Output S
```

Exercise 1

a) $S = 1 + 2 + 3 + \dots + n$

For the second case, we do as in class. Note that we need define S and use a loop.

```
Input n
        S:=0                        //for the sum, we first assign S to be zero
        for i:=1 to n do
                S:=S+i              //we continuously compute partial sum
        end for
Return S
```

Exercise 1

b) $S = 1^2 + 2^2 + . . . + (n-1)^2 + n^2$

It can be done similarly as in (a).

```
Input n
        S:=0
        for i:=1 to n do
                S:= S + i²
        end for
Return S
```

Exercise 2

Write a pseudocode of converting decimal numbers to their binary representations.

It is a bit hard in this exercise. You may want to look at here for more information about binary representations of demical numbers

https://www.bottomupcs.com/chapter01.xhtml

And more on the algorithm in this exercise https://indepth.dev/the-simple-math-behind-decimal-binary-conversion-algorithms/

https://runestone.academy/runestone/books/published/pythonds/BasicDS/ConvertingDecimalNumberstoBinaryNumbers.html

Exercise 2

Basically, any number x can be written uniquely in the following form

$$x = a_0 + a_1 2 + a_2 2^2 + \cdots + a_k 2^k$$

Where $a_k = 1, a_0,\ldots, a_{k-1} \in \{0,1\}$

Then we write $a_k a_{k-1} \ldots a_1 a_0$ to be the binary representation of x

Example:

3 = 1 + 2, and so a_0=1 and a_1=1, so the binary representation of 3 is 11

7 = 1 + 2 +2^2 and so its binary representation is 111

6 = 2 +2^2 and so its binary representation is 110

Exercise 2

Now we want to find $a_0,\ldots, a_k \in \{0,1\}$ such that

$$x = a_0 + a_1 2 + a_2 2^2 + \cdots + a_k 2^k$$

Note that if x is even then a_0=0, and if x is odd then a_0=1.

Hence, given x, it is easy to find a_0 depending on its parity

So we can easily get a_0 = x mod 2

After getting a_0, note that

$$\frac{x - a_0}{2} = a_1 + a_2 2^1 + \cdots + a_k 2^{k-1}$$

And so, depending on the parity of $\frac{x-a_0}{2}$, we can find a_1. We then just continue the process to find all the $a_0, a_1, \ldots, a_k$

Detail then can be found at here: https://chortle.ccsu.edu/AssemblyTutorial/zAppendixH/appH_4.html

Exercise 2

Now we want to find a_0,…, $a_k \in \{0,1\}$ such that

$$x = a_0 + a_1 2 + a_2 2^2 + \cdots + a_k 2^k$$

Note that if x is even then a_0=0, and if x is odd then a_0=1.

Hence, given x, it is easy to find a_0 depending on its parity

So we can easily get a_0 = x mod 2

After getting a_0, note that

$$\frac{x - a_0}{2} = a_1 + a_2 2^1 + \cdots + a_k 2^{k-1}$$

And so, depending on the parity of $\frac{x-a_0}{2}$, we can find a_1. Note that using math notation, we can write $\frac{x-a_0}{2}$ as x div 2

We then just continue the process to find all the $a_0, a_1, \ldots, a_k$

Exercise 3

Write a pseudocode to compute the LCM (least common multiple) of two numbers.

Hint: note that for two numbers A and B, we have

$$AB = LCM(A,B) \times GCD(A,B)$$

And hence $LCM(A,B) = AB/GCD(A,B)$

You learnt how to compute GCD(A,B) in class, so just need to do one more step to get LCM(A,B)

Exercise 3

```
Input A,B
        S:=AB                    //compute the product of A and B
        If A < B, swap(A,B)
        While B is not equal to 0
                r = A mod B
                A = B
                B = r
        End While                //after this, B will be GCD of original A and B
        result: = S/B            //this is exactly AB/GCD(A,B)
Output result
```

Exercise 4

Write pseudocode to compute the power of a number: a^n

Hint: you can do similarly as in Exercise 1. Note that here you compute the product n times, not the sum as in Exercise 1. The difference is the following:

- For computing sum in Exercise 1, we use S, and start with S:=0
- For computing product, we use P, and start with P:=1

Exercise 4

```
Input a, n
        P:=1
        for i:=1 to n do        //we multiply n times corresponding to n power
                P:=P x a        //continuously multiply the product with a
        end for
Return P
```