# Recursion

## 1 Introduction

Recursion is a problem solving technique used to solve a problem that can be expressed in smaller instances of the same problem. Creating recursive solutions often rely to finding the *recurrence relation* of the problem.

**EXAMPLE!**
A simple example of a recursive problem is summation and is defined as follows:

Let $S(n)$ be the sum of all numbers from 1 to n. This can be written down as:

$$S(n) = 1 + 2 + 3 + ... + (n - 1) + n$$

From the definition above, we can see that the summation problem can be expressed in smaller instances of itself. Removing $n$ from the equation, we will have the sum of all the numbers up to $n-1$, which is just the summation function but with a different parameter, as shown below:

$$\begin{gathered} S(n) = [1 + 2 + 3 + ... + (n - 1)] + n \\ S(n) = S(n - 1) + n \end{gathered}$$

Since we are able to represent the problem using a smaller version of itself, we now have its recurrence relation.

$$S(n) = S(n - 1) + n$$

From this recurrence relation, we can solve for the answer to the problem by *expanding the equation until we arrive at the simplest subproblem.*

$$\begin{aligned} S(n) &= S(n - 1) + n \\ S(n) &= S(n - 2) + (n - 1) + n \\ S(n) &= S(n - 3) + (n - 2) + (n - 1) + n \\ S(n) &= S(1) + 2 + ... + (n - 1) + n \end{aligned}$$

Once we arrive at $S(1)$, which cannot be further expressed into a smaller subproblem, we arrive at a **base case** which terminates our recursive function.

### 1.1 Writing Recursive Functions

Based on the example above, the problems that have recursive solutions must have two parts:

1. **Recursive case**
   - the relationship of the bigger problem to its smaller subproblems
   - will determine how the problem will reach its base case
   - this is the step that is called repeatedly
2. **Base case** (a.k.a. terminating condition)
   - The smallest subproblem that must be reached by the recursive case

### 1.2 Recursion in Programming

In programming, recursion is a technique that allows the programmer to express processes in terms of itself. **Basically, it is just a function calling itself**. In a way, recursive functions are just functions whose processes are repeating itself to get the answer.

So now, how will we express the summation problem in our program using recursion? You need to take note of the *recursive case* and the *base case.* These cases will be the condition inside your recursive function.

```
int summation(int n) {
    if( n == 1 ) { return 1; } //this is our base case: S(1) = 1
    else {                     //this is the recursive case:
        return (summation(n-1) + n); //S(n) = S(n-1) + n
    }
}
```

## 2 Header Files

Header files in C are files that contain several C function definitions or declarations. These header files can be included to a C program to be able to utilize its built in functions and for programmers to easily manage their code.

These header files also encourage modularity and reusability of the code. Creating header files is as easy as defining functions in your source code, except the definitions are contained in a separate file.

There are two types of header files:

1. **Built-in header files**
   Built-in header files contains built-in functions that comes together with the `GNU C Compiler (gcc)`. The `stdio.h` you type in your program is a built-in header file in C which contains many built-in functions (`printf` and `scanf` are just some of them).

   There are more built in functions in other C header libraries such as `time.h`, `stdlib.h`, and so on. We will discuss some of these header files in the future.

   To add a header file to your C program, you need to follow the syntax below:

   ```
   #include <header_file_name.h>
   ```

2. **User-defined header files**
   User-defined header files are header files that are created by the programmer and contains user-defined functions, declarations, and/or variables. Below is the syntax to include a user-defined header file:

   ```
   #include "header_file_name.h"
   ```

**NOTE:**
You use angled brackets (`< >`) if you are adding a built-in header file.
Use double quotes (`" "`) instead if you are adding a user-defined header file.