

## Lab03

All functions should be in a main.cpp file. Only include: **iostream**, **string**, **cmath**, **cstdlib**, **vector**, and **stdexcept**.

### Task 1: (1 point)

Create a function called `summation()` that takes an integer `n` as input and returns the sum of all integers from 1 to `n` using recursion.

Example: `summation(10) => 55`

### Task 2: (1 points)

Create a function called `summationConstant()` that takes an integer `n` as input and returns the sum of all integers from 1 to `n`. The runtime for this function should be constant time  $O(1)$ . (Hint: use a formula)

Example: `summationConstant(10) => 55`

### Task 3: (2 points)

Create a function called `digit()` that takes a character `c` as input and returns its integer value if it is a digit (0-9) or a letter (A-F, a-f) representing a hexadecimal digit. (A=10, B=11, C=12, D=13, E=14, F=15). If the character is not a valid digit or letter, the function should throw an exception.

Example: `digit('3') => 3`

Example: `digit('A') => 10`

### Task 4: (3 points)

Create a function called `weightedSum()` that takes a constant string reference parameter `str` and an integer `base` as input. The function should compute the weighted sum of the digits in the string based on their position and the given base using recursion. Use the `digit()` function from task 3. Multiply the result of `digit()` with the base raised to the power of the position.

Example:  $01010 = (0 \cdot 2^4) + (1 \cdot 2^3) + (0 \cdot 2^2) + (1 \cdot 2^1) + (0 \cdot 2^0)$   
 $= 0 + 8 + 0 + 2 + 0$   
 $= 10$

Example: `weightedSum("01010", 2) => 10`

### Task 5: (3 points)

Create a generic function called `binarySearchR()` that implements the recursive version of the binary search algorithm.

Example: `binarySearchR({1,2,3,4}, 2, 0, 3) => 1`