Lab 3: Recursion

UWYO COSC 2030

1 Introduction to Recursion

Recursion is the process by which a function calls itself directly or indirectly until reaching a criterion for termination. For example, consider the following recursive function for calculating the factorial of a given integer:

```cpp
//function to recursively calculate the factorial of a given number
long int factorial(long int val) {
    if(val == 0) {
        return 1;
    }
    else {
        return val * factorial(val - 1);
    }
}
```

The first condition is known as the base case, or the case in which the solution is known or trivial. The base case provides the stopping condition for recursion and prevents the program from running infinitely. In this example, the base case occurs when `val == 0`, as $0! = 1$. The second condition is known as the recursive case. In the recursive case, the problem is divided into smaller subproblems and the function is called recursively on the subproblems. In this example, the function calls itself to compute the factorial of `val - 1`). In the process of recursion, the solutions to subproblems in successive calls are combined to yield the solution to the original problem. It is important to ensure that a recursive function actually reaches the base case and terminates rather than entering an infinite loop.

In summary, the general approach to recursion is as follows:

- Define a base case
- Define a recursive case
- Ensure the recursion terminates
- Combine the solutions

For additional resources, please consult:

- `https://www.geeksforgeeks.org/dsa/introduction-to-recursion-2/`
- `https://en.wikipedia.org/wiki/Recursion`
- `https://www.w3schools.com/cpp/cpp_functions_recursion.asp`

2 Recursion Example

You can think of the output of a recursive function as a nested mathematical statement. For example, the output of the function below:

```cpp
//this code returns a string that counts down from the input number
string printCountdown(int num)
{
  if(num == 0)
  {
```

Edited by Milan Wolff (based on labs by Michael Stoll and Justin Wang for UWYO COSC 2030)

```
    return "0. Blastoff!";
  }
  else
  {
    return (to_string(num) + ", ") + printCountdown(num-1);
  }
}
```

can be thought of as (5, + (4, + (3, + (2, + (1, + (0. Blastoff!)))))) where each ( ) is a function call of printCountdown().

Each time a recursive call is made, the current return statement pauses until that recursive function call is resolved. Once it reaches the exit statement/base case (which doesn't have a recursive call), it is able to resolve and finish all of the previous return statements that were on hold. The final output string of printCountdown(5) would be "5, 4, 3, 2, 1, 0. Blastoff!"

3 Assignment

Github Classroom Assignment Link: `https://classroom.github.com/a/TWrrFHgY`

Using the Github classroom link, accept the assignment. Read the instructions in the C++ program Lab3.cpp and complete the following functions using recursion:

- findBinaryNotation
- isPalindrome
- recurSummation
- arrRev

Do not change `main()` or the function definitions, simply navigate to the first function declaration and begin there. Remember that the most important part of recursion is knowing your exit condition (i.e. when do you stop?). You may code in whatever editor you want (Visual Studio, VS Code, Nano, Terminal, etc.) or use an online code editor/compiler like OnlineGDB.

4 Submission

Turn in on Github by uploading only the .cpp file. Make sure you include a readme with your name, the lab section you attended, and any help you gave/received (if applicable). Submissions missing these requirements will be penalized.