# COP 3223 Assignment #3

## Program A: Hello Again! (helloagain.c):

By now, it should be old-hat to write Hello World. So lets kick it up a notch. Write a program that prompts the user for a positive integer *N*, then outputs "Hello!" every even step (0 indexed), and "Hello Again!" every odd step, *N* times, one line each.

**Sample Run (User input in bold and italics):**

```
How many times would you like me to repeat it?
5
Hello!
Hello Again!
Hello!
Hello Again!
Hello!
```

## Program B: Factor Game (factorgame.c):

For this program, lets write a game. This game will have 2 users, who will alternate playing. Player 1 starts, followed by player 2, followed by player 1, etc. until someone loses.

On their turn, a player may take any factor that divides current number evenly. "Taking" this factor will divide the currently number by that, modifying it. For the sake of avoiding stalemate, we will assume that 1 is not a factor of any number. So, at each turn, output who's turn it is, what the current number is, and prompt the user to input that player's factor. If the input number is not a factor of the current number or is less than 2, reprompt the user (without changing turns) until it is valid. The game ends (and the program should end) when the current number is 1 (in other words, when there is no factor left to take). The player who took the last factor wins. When the game ends, output who won.

For ease of testing, instead of randomly generating a starting number, we will simply prompt the user for the starting number when the program starts. However, you should write the program to manage all modifications (dividing factors) of this number from that point forward.

**Sample Run (User input in bold and italics):**

```
What number should the game be played with?
```
***4***
```
The current number is 4. Player 1, enter a factor.
```
***2***
```
The current number is 2. Player 2, enter a factor.
```
***2***
```
Game over. Player 2 won!
```

**Sample Run (User input in bold and italics):**

```
What number should the game be played with?
12
The current number is 12. Player 1, enter a factor.
2
The current number is 6. Player 2, enter a factor.
3
The current number is 2. Player 1, enter a factor.
2
Game over. Player 1 won!
```

**Sample Run (User input in bold and italics):**

```
What number should the game be played with?
10
The current number is 10. Player 1, enter a factor.
3
Invalid factor. 3 does not evenly divide 10. Enter a valid factor.
The current number is 10. Player 1, enter a factor.
2
The current number is 5. Player 2, enter a factor.
1
Invalid factor. Numbers less than 2 are not allowed.
The current number is 5. Player 2, enter a factor.
5
Game over. Player 2 won!
```

Note: you may assume factor guesses from the user will always be integers. But within that bound, you need to make sure your program handles the different possible invalid integer inputs, not just valid factors. The two associated output messages are in the above example.

## Program C: Evergreen Tree (tree.c):

Let's get a bit artistic. We're going to draw a simplistic evergreen tree with ASCII art. We'll use * for the pines and # for the trunk.

To help us draw the tree, we'll ask the user for two integers. The first will be a positive integer: the number of levels in the tree. The second will be a positive integer greater than 1: the number of layers in each level.

We draw an evergreen tree by starting with a layer of width 1. Each time we move to the next layer, we will increase its width by 2 (one pine added to each side to keep it symmetrical).

When we move down to the next layer, we decrease this count by 4 (removing two pines from each side) and continue this process. Once we have run out of levels, we finish it with a width 3, height 4 trunk (centered beneath the tree). Use the samples for reference if anything isn't clear.

Note: make sure you use the correct number of spaces before each line to get the tree to line up correctly. The longest line(s) of the tree should have no spaces before them, and any shorter lines should have the correct number to keep the lines centered relative to that.

Hint: don't forget that \n is not required in printf, it is only needed when you want to put a line-break (like pressing enter) at that point. Also, remember you can write a printf of only "\n" if needed. If this is not clear, imagine that the computer types whatever you put between the quotation marks, in order, character by character, except that when it sees \n it instead presses the <Enter> button (and except when it sees the % character, as we've learned before, but that isn't relevant here).

Hint 2: To make it easier to know how many spaces to put before each line, you may want to first calculate the maximum width of any line in that tree. This can be done with either math or with a separate loop.

**Sample Run (User input in bold and italics):**

```
How many levels should this tree have?
1
How many layers should each level have?
6
     *
    ***
   *****
  *******
 *********
***********
    ###
    ###
    ###
    ###
```

**Sample Run (User input in bold and italics):**

```
How many levels should this tree have?
4
How many layers should each level have?
6
                   *
                  ***
                 *****
                *******
               *********
              ***********
               *********
              ***********
             *************
            ***************
           *****************
          *******************
           *****************
          *******************
         *********************
        ***********************
       *************************
     ***************************
       *************************
     ***************************
    *****************************
  *******************************
 *********************************
***********************************
                  ###
                  ###
                  ###
                  ###
```

**Sample Run (User input in bold and italics):**

```
How many levels should this tree have?
6
How many layers should each level have?
2
 *
***
 *
***
 *
***
 *
***
 *
***
 *
***
###
###
###
###
```

Note: some operating systems limit the number of characters per column. If you create an input that outputs more than 80 characters in a line, your tree layers might wrap to the next line. This may cause trees that are too large to look broken.

## Deliverables:

Please submit three separate .c files for your solutions to these problems via WebCourses by the designated due date:

Program A: **helloagain.c**
Program B: **factorgame.c**
Program C: **tree.c**