**[\*]Q1. Printing Digits of a Number**

Objectives: Manipulate int variables by using the modulo operator

Q: Write a program that asks the user for a 4 digit number, then prints the sum of the digits.

|  |
| --- |
| Enter n: 4560  4 + 5 + 6 + 0 = 15 |

**[\*]Q2. Leap year**

Objectives: Combine conditionals and logical short-circuiting

Q: Write a program that prompts the user for a year, and then prints whether it is a leap year or not. Leap years are years that can be evenly divided by 4, except for those years which are divisible by 100. Exceptions to the latter rule are years that are divisible by 400 - they are leap years though they are also divisible by 100

|  |
| --- |
| Enter year: 2020  2020 is a leap year |
| Enter year: 1900 1900 is not a leap year |
| Enter year: 2000 2000 is a leap year |

**[\*\*]Q3. Longest streak sum**

Objectives: Use while-loops and store important information using declared variables

Q: Write a program that continually prompts the user for integers, until the user enters a negative number. The program then displays the *sum* of longest streak of increasing numbers that the user entered.

|  |
| --- |
| Enter n: 4  Enter n: 2 Enter n: 7  Enter n: 5  Enter n: 1  Enter n: 2  Enter n: 3  Enter n: -1 Longest streak sum: 6 |
| Enter n: -1  Longest streak sum: 0 |

**[\*]Q4. Number guesser**

Objectives: Use while loops, store result variables, and use conditional statements with comparison

Q: Write a program that generates a random number from 1-100, which the user then has to guess. The program will tell the user if their guess is too high, too low, or out of bounds. Once the user has guessed correctly, the program will print out the number of guesses taken.

Here is a sample run (your results may vary):

|  |
| --- |
| Enter n: 60 Too low!  Enter n: 110 Out of bounds! Enter a number from 1 to 100  Enter n: 85 Too high!  Enter n: 78 Jackpot! 4 guesses taken |

**[\*\*]Q5. Summing Harmonic Series**

Objectives: Loop and avoid errors involving datatypes and type-casting

Write a program in C to display the first nth terms of the [harmonic series](https://en.wikipedia.org/wiki/Harmonic_series_(mathematics)) and their sum

1 + 1/2 + 1/3 + ... + 1/n

|  |
| --- |
| Input the number of terms: 3  1/1 + 1/2 + 1/3 = 1.8333 |

**[\*\*]Q6. Inverted Hollow Triangle**

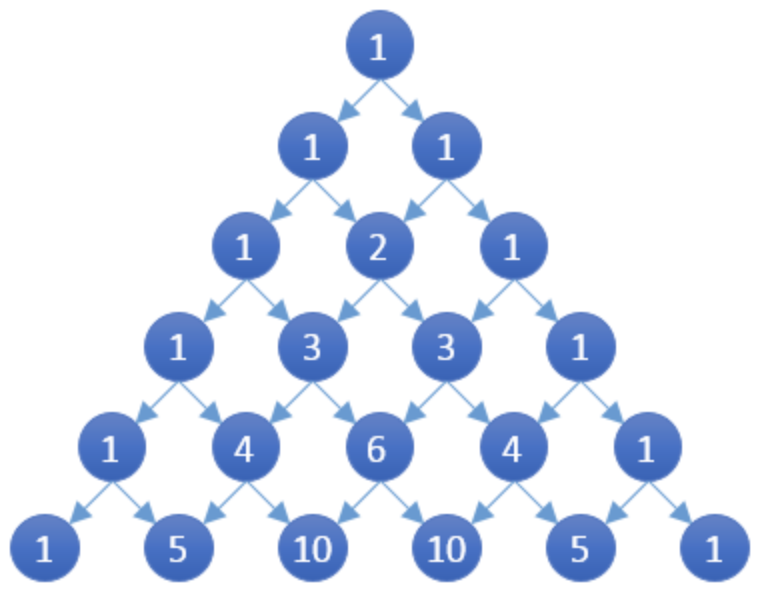
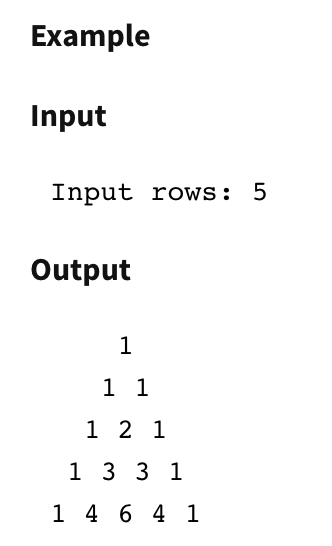
Objective: Apply pattern printing techniques to a new problem, understanding how nested for-loops work and how to apply them

Prompt for an integer n, then print an inverted hollow triangle with n rows.

|  |
| --- |
| Enter n: 3  #####  # #  #  Enter n: 4  #######  # #  # #  # |

**[\*\*]Q7. Pascal Triangle**

Write a function that prints the first n rows of a pascal triangle



To find the nth term of a pascal’s triangle, we can use the following formula:

Where n is the row number and k is the term of that row

|  |
| --- |
| Enter n: 2  1  1 1  Enter n: 5  1  1 1  1 2 1  1 3 3 1  1 4 6 4 1 |

**[\*\*]Q8. Find all factors**

Objective: Use loops and logical conditions in an efficient manner

Prompt for an integer n. Print all factor pairs for that number, starting from the largest factor. Do not repeat results. (For example, factor pairs (4, 3) and (3, 4) are considered the same)

|  |
| --- |
| Input: 60  Factors: (60, 1), (30, 2), (20, 3), (15, 4), (12, 5),  (10, 6)  Input: 0  Factors: ()  Input: -20  Factors: (20, 1), (10, 2), (5, 4) |

**[\*\*]Q9. Collatz Conjecture**

Objective: Use recursion.

The [collatz conjecture](https://en.wikipedia.org/wiki/Collatz_conjecture) is a sequence of operations that takes in any positive integer, and eventually ends with the value of 1. It is defined as follows:

* Start with any integer, *n.*
* If n is 1, stop the sequence
* If n is even, the next number in the sequence is n/2. Otherwise if n is odd, the next number in the sequence is (n\*3) + 1.

You are to write a *recursive* function, collatz(), which takes in a positive integer and returns the number of steps it takes for the sequence to reach 1.

|  |
| --- |
| Function call: collatz(1)  Expected return value: 0  Function call: collatz(5)  Expected return value: 5  Function call: collatz(7)  Expected return value: 16 |

**[\*\*]Q10. Swap array values**

Write a function, swap\_values\_apart(), that takes in an array (arr), the size of the array (size) and a distance (n). It swaps array values that are n spaces apart. Array elements that do not have a pair complement are unchanged.

Given an array as {4, 12, 6, 13, 1, 0}, swap\_values\_apart(arr, size, 3) would result in an array of {13, 1, 0, 4, 12 , 6}

Given an array as {5, 2, 8, 9, 10}, swap\_values\_apart(arr, size, 4) would result in an array of {10, 2, 8, 9, 5}

Given an array as {3, 1, -3}, swap\_values\_apart(arr, size, 5) would result in an array of {3, 1, -3}

**[\*\*\*]Q11. Ascending number pairs**

Write a function, print\_ascending\_pairs(), that takes in an integer and prints out its 2-digit number pairs in ascending order.

|  |
| --- |
| Function call: print\_ascending\_pairs(709)  Expected output: 09, 70  Function call: print\_ascending\_pairs(12345)  Expected output: 12, 23, 34, 45  Function call: print\_ascending\_pairs(1)  Expected output: none |