Lab 1

All functions should be in a main.cpp file. Only include: `Util.h`, `iostream`, `string`, `cmath`, and `stdexcept`.

Helper Functions: `string pad(const string&, int)` => pads a binary string to a specific length with its sign bit.

`string trim(const string&)` => removes padded sign bits until there is only one

`string onesComplement(const string&)` => inverts each bit to its one's complement

Task 1: (1 point)

Create a function called **binaryToDecimal()** that takes a constant string reference parameter. The string parameter represents a signed binary number. Return an integer that represents the binary number as a decimal number. If the number is negative, DO NOT USE two's complement. Perform the weighted sum technique. However, instead of adding the most significant bit's weight to the accumulated sum, subtract it. This will work on positive and negative binary numbers. Ensure your string parameter consists of only 0s and 1s.

Example: $\mathbf{111010_2} = -(1 \text{ x } 2^5) + (1 \text{ x } 2^4) + (1 \text{ x } 2^3) + (0 \text{ x } 2^2) + (1 \text{ x } 2^1) + (0 \text{ x } 2^0)$

$= (1 \text{ x } 2^4) + (1 \text{ x } 2^3) + (0 \text{ x } 2^2) + (1 \text{ x } 2^1) + (0 \text{ x } 2^0) - (1 \text{ x } 2^5)$

$= 16 + 8 + 0 + 2 + 0 – 32$

$= -6$

Example: $001010_2 = -(0 \text{ x } 2^5) + (0 \text{ x } 2^4) + (1 \text{ x } 2^3) + (0 \text{ x } 2^2) + (1 \text{ x } 2^1) + (0 \text{ x } 2^0)$

$= (0 \text{ x } 2^4) + (1 \text{ x } 2^3) + (0 \text{ x } 2^2) + (1 \text{ x } 2^1) + (0 \text{ x } 2^0) - (0 \text{ x } 2^5)$

$= 0 + 8 + 0 + 2 + 0 – 0$

$= 10$

Task 2: (1.5 points)

Create a function called **add()** that takes two constant string reference parameters. The string parameters represent signed binary numbers. Return a string that represents the binary number sum of adding the two parameters. Ensure your string parameter consists of only 0s and 1s. Use the provided pad() function.

Example: `add("01010", "01100") -> "010110" (Carry-out becomes sign bit)`

Example: `add("01010", "11100") -> "00110" (Carry-out ignored)`

Task 3: (0.5 points)

Create a function called **twosComplement()** that takes a constant string reference parameter. Using the provided onesComplement function, return the two's complement of the string parameter.

Example: `twosComplement("1001001") -> "0110111"`

Task 4: (0.5 points)

Create a function called **subtract()** that takes two constant string reference parameters. The string parameters represent signed binary numbers. Return a string that represents the binary number difference of subtracting the second parameter from the first. Remember, subtraction is just addition after changing the sign of the subtrahend. 10 – 5 = 10 + (-5)

Example: `subtract("01010", "01100") -> "11110"`

Task 5: (1.5 points)

Create a function called **decimalToBinary()** that takes an integer parameter. Convert the signed integer to its equivalent signed binary number.

Example: `decimalToBinary(-55) -> "1001001"`