C Assignment

TCSS 380 – Spring 2024

## Description

Continuing your work from before, your organization needs the implementation of the binary function converted into C. You are assigned to create a new c program that implements multiple functions that perform binary operations.

## Requirements

### Implementation

Create a new c file called **binaryOps.c**. I will use Unit testing while grading your submission. As such, you must precisely follow the API described in the next section.

### Notes

* This representation of binary numbers:
  + does not account for negative values. Do not consider a signed bit or other negative representations.
  + uses the array **[0]** to represent the value for 0.
  + only considers integers.

### Unit Tests

In the main function, write enough informal unit test cases to gain a high level of confidence that your solution is correct. Label each test case with a comment and print to the console when run. Make sure you check for edge cases, especially **[0]**!

### API

**binNegate(int \*bin, int size, int \*outSize)**

Performs and returns logical negation on each bit, forming the ones' complement of the given binary value and returns the result. Bits that are 0 become 1, and those that are 1 become 0.

**Parameters:**

**bin** – a binary number represented as an array of **int** containing the individual bits in order. All leading (left) 0s are ignored except for **[0]**.

**size** – the number of bits represented in this array. This size will include leading (left) 0s.

**outSize** – (OUT Parameter) the number of bits represented in the resulting array.

Example:

**[1,0,0,1,1,1,0,1,1] [1] [0]**

**Returns:**

the negation of the argument as a binary number represented as an array of **int** containing the individual bits in order. The resulting list must not contain leading (left) 0s except for **[0]**.

Example:

**[1,1,0,0,0,1,0,0] [0] [1]**

**binAnd(int \*bin1, int size1, int \*bin2, int size2, int \*outSize)**

Performs and returns the logical AND operation on each pair of the corresponding bits and returns the result. Thus, if both bits in the compared position are 1, the bit in the resulting binary representation is 1 otherwise, the result is 0.

**Parameters:**

**bin1** – a binary number represented as an array of **int** containing the individual bits in order.

**size1** – the number of bits represented in the array **bin1**. This size will include leading (left) 0s.

**bin2** – a binary number represented as an array of **int** containing the individual bits in order.

**size2** – the number of bits represented in the array **bin2**. This size will include leading (left) 0s.

**outSize** – (OUT Parameter) the number of bits represented in the resulting array.

Example:

**[1,0,0,1,1,1,0,0,1] [1,1,0,0]**

**Returns:**

the logical AND operation on each pair of the corresponding bits between each argument as an array of **int** containing the individual bits in order. The resulting list must not contain leading (left) 0s except for **[0]**.

Example:

**[1,0,0,0]**

**binOr(int \*bin1, int size1, int \*bin2, int size2, int \*outSize)**

Performs and returns the logical OR operation on each pair of the corresponding bits and returns the result. Thus, if either bits in the compared position are 1, the bit in the resulting binary representation is 1 otherwise, the result is 0.

**Parameters:**

**bin1** – a binary number represented as an array of **int** containing the individual bits in order.

**size1** – the number of bits represented in the array **bin1**. This size will include leading (left) 0s.

**bin2** – a binary number represented as an array of **int** containing the individual bits in order.

**size2** – the number of bits represented in the array **bin2**. This size will include leading (left) 0s.

**outSize** – (OUT Parameter) the number of bits represented in the resulting array.

Example:

**[1,0,0,1,1,1,0,0,1] [1,1,0,0]**

**Returns:**

the logical OR operation on each pair of the corresponding bits between each argument as an array of **int** containing the individual bits in order. The resulting list must not contain leading (left) 0s except for **[0]**.

Example:

**[1,0,0,1,1,1,1,0,1]**

**binXor(int \*bin1, int size1, int \*bin2, int size2, int \*outSize)**

Performs the logical XOR operation on each pair of the corresponding bits and returns the result. The result in each position is 1 if only one of the bits is 1, otherwise, the result is 0.

**Parameters:**

**bin1** – a binary number represented as an array of **int** containing the individual bits in order.

**size1** – the number of bits represented in the array **bin1**. This size will include leading (left) 0s.

**bin2** – a binary number represented as an array of **int** containing the individual bits in order.

**size2** – the number of bits represented in the array **bin2**. This size will include leading (left) 0s.

**outSize** – (OUT Parameter) the number of bits represented in the resulting array.

Example:

**[1,0,0,1,1,1,0,0,1] [1,1,0,0]**

**Returns:**

the logical XOR operation on each pair of the corresponding bits between each argument as an array of **int** containing the individual bits in order. The resulting list must not contain leading (left) 0s except for **[0]**.

Example:

**[1,0,0,1,1,0,1,0,1]**

**binAdd(int \*bin1, int size1, int \*bin2, int size2, int \*outSize)**

Performs an addition operation (a + b) on the arguments and returns the result.

**Parameters:**

**bin1** – a binary number represented as an array of **int** containing the individual bits in order.

**size1** – the number of bits represented in the array **bin1**. This size will include leading (left) 0s.

**bin2** – a binary number represented as an array of **int** containing the individual bits in order.

**size2** – the number of bits represented in the array **bin2**. This size will include leading (left) 0s.

**outSize** – (OUT Parameter) the number of bits represented in the resulting array.

Example:

**[1,1,0,0,1] [1,1,0,0]**

**Returns:**

the result of performing an addition (a + b) operation of the arguments as an array of **int** containing the individual bits in order. The resulting list must not contain leading (left) 0s except for **[0]**.

Example:

**[1,0,0,1,0,1]**

**trimLeading(int \*bin, int size, int \*outSize)**

Returns the result of removing all leading(left) 0s from the argument.

**Parameters:**

**bin** – a binary number represented as an array of **int** containing the individual bits in order. All leading (left) 0s are ignored except for **[0]**.

**size** – the number of bits represented in this array. This size will include leading (left) 0s.

**outSize** – (OUT Parameter) the number of bits represented in the resulting array.

Example:

**[0,0,0,1,1,1,0,0,1]**

**Returns:**

the result of removing all leading(left) 0s from the argument as an array of **int** containing the individual bits in order. The resulting list must not contain leading (left) 0s except for **[0]**.

Example:

**[1,1,1,0,0,1]**