CMPT 130 – FIC 202203 – Assignment 2

Due Date and Time: Tuesday 8 November 2022 at 11:55PM

Instructor: Dr. Yonas Tesfazghi Weldeselassie (Ph.D.)

Read this document in its entirety and carefully before you start anything and understand it. If you have any questions, don't hesitate to email me.

Problem Statement

In this assignment, we will work with numeric information representation using unsigned binary representation, sign and magnitude representation, and two's complement representation. In order to store binary bits, we will use partial arrays with pre-specified maximum size but then work with our choice of bit pattern length that does not exceed the pre-specified maximum size.

You are given the following test program that contains a main function and some supporting functions.

```
#include <iostream>
#include <cmath>
using namespace std;
int getBitPatternLength(const int array_size)
{
	int bit_pattern_length;
	cout << "Enter the required bit pattern length between 1 and " << array_size << ": ";
	cin >> bit_pattern_length;
	while (bit_pattern_length < 1 || bit_pattern_length > array_size)
	{
		cout << "You must enter between 1 and " << array_size << ". Enter again please: ";
		cin >> bit_pattern_length;
	}
	return bit_pattern_length;
}
void printArray(const int arr[], const int bit_pattern_length)
{
	for (int i = 0; i < bit_pattern_length; i++)
		cout << arr[i];
}
int selectComputation()
{
	cout << "Select your computation" << endl;
	cout << "   1. Unsigned Binary Representation Computation" << endl;
	cout << "   2. Sign and Magnitude Representation Computation" << endl;
	cout << "   3. Two's Complement Representation Computation" << endl;
	cout << "   4. Exit Program" << endl;
	int selection;
	cout << "Enter your selection (1, 2, 3, or 4): ";
	cin >> selection;
	while (selection != 1 && selection != 2 && selection != 3 && selection != 4)
	{
		cout << "Please enter a correct choice: ";
		cin >> selection;
	}
	return selection;
}
int main()
{
	cout << "This program demonstrates numeric information representation using" << endl;
	cout << " *** Unsigned Binary Representation" << endl;
	cout << " *** Sign and Magnitude Binary Representation" << endl;
	cout << " *** Two's Complement Binary Representation" << endl << endl;
	cout << "In addition, the program demonstrates" << endl;
	cout << " *** Two's complement binary addition, and" << endl;
	cout << " *** Conversion from two's complement to decimal." << endl << endl;
	const int array_size = 32;
	do
	{
		int selection = selectComputation();
		if (selection == 1)
		{
			int A[array_size];
			int bit_pattern_length = getBitPatternLength(array_size);
			int num;
			cout << "Enter a non-negative integer: ";
			cin >> num;
			while (num < 0)
			{
				cout << "You must enter a non-negative integer. Enter again please: ";
				cin >> num;
			}
			computeUnsignedBinary(A, bit_pattern_length, num);
			cout << "The unsigned binary representation of " << num << " in " <<
bit_pattern_length << " bit is ";
			printArray(A, bit_pattern_length);
			cout << endl;
		}
		else if (selection == 2)
		{
			int A[array_size];
			int bit_pattern_length = getBitPatternLength(array_size);
			int num;
			cout << "Enter an integer: ";
			cin >> num;
			computeSignAndMagnitudeBinary(A, bit_pattern_length, num);
			cout << "The sign and magnitude binary representation of " << num << " in " <<
bit_pattern_length << " bit is ";
			printArray(A, bit_pattern_length);
			cout << endl;
		}
		else if (selection == 3)
		{
			int A1[array_size], A2[array_size], A3[array_size];
			int bit_pattern_length = getBitPatternLength(array_size);
			int num1, num2;
			cout << "Enter an integer: ";
			cin >> num1;
			computeTwosComplementBinary(A1, bit_pattern_length, num1);
			cout << "The two's complement binary representation of " << num1 << " in " <<
bit_pattern_length << " bit is ";
			printArray(A1, bit_pattern_length);
			cout << endl;
			cout << "Enter a second integer: ";
			cin >> num2;
			computeTwosComplementBinary(A2, bit_pattern_length, num2);
			cout << "The two's complement binary representation of " << num2 << " in " <<
bit_pattern_length << " bit is ";
			printArray(A2, bit_pattern_length);
			cout << endl;
			binaryAddition(A1, A2, bit_pattern_length, A3);
			cout << "The binary sum of ";
			printArray(A1, bit_pattern_length);
			cout << " and ";
			printArray(A2, bit_pattern_length);
			cout << " is ";
			printArray(A3, bit_pattern_length);
			cout << endl;
```

```
                        int sum = twosComplementBinaryToDecimal(A3, bit_pattern_length);
                        cout << "The integer value of the binary sum is " << sum << endl;
                        if (sum == num1 + num2)
                                cout << "This is a correct result." << endl;
                        else
                                cout << "This is not correct result because our bit pattern is too small to
store the result." << endl;
                }
                else
                        break;
                system("Pause");
                cout << endl << endl;
        }while (true);

        system("Pause");
        return 0;
}
```

Your aim is to implement the missing functions so that the given test program runs without any syntax, runtime, or semantic errors. The missing functions are described below.

1. **`computeUnsignedBinary function`**
   This function takes a static array, bit pattern length, and a non-negative integer number as its arguments. The function must populate the array with the unsigned binary representation of the non-negative integer number argument in the given bit pattern length. Assume the all the arguments are valid.

2. **`computeSignAndMagnitudeBinary function`**
   This function takes a static array, bit pattern length, and an integer number as its arguments. The function must populate the array with the sign and magnitude binary representation of the integer number argument in the given bit pattern length. Assume the all the arguments are valid.

   The sign bit of a zero integer number must be set to 0 (not 1), see the sample run outputs below.

3. **`computeTwosComplementBinary function`**
   This function takes a static array, bit pattern length, and an integer number as its arguments. The function must populate the array with the two's complement binary representation of the integer number argument in the given bit pattern length. Assume the all the arguments are valid.

4. **`binaryAddition function`**
   This function takes two static arrays, bit pattern length, and a third static array as its arguments. The function must populate the third array with the binary sum of the first two arrays in the given bit pattern length. Assume the first two arrays are populated with binary bits in the given bit pattern length. Moreover assume that all the arguments are valid.

5. **`twosComplementBinaryToDecimal function`**
   This function takes a static array and bit pattern length as its arguments. The function must return the decimal (integer) value computed from the two's complement binary representation given in the array argument in the given bit pattern length. Assume the array is populated with binary bits in the given bit pattern length and that it is a two's complement representation.

You may define additional supporting functions. If you do then you must include your supporting functions in your submission.

Please note that any overflow bit during your computations must be ignored. For example when you represent a given non-negative integer number in unsigned binary representation say for example in 5 bit but then the integer number requires more than 5 bits to represent it correctly, then only 5 bits should be stored in the array and the remaining over flow bits must be ignored. Similarly any overflow bit during arithmetic in two's complement representation must be ignored.

Restriction

You are not allowed to add any additional include directive or namespace in your program except for the given include directives and namespaces. Moreover you are not allowed to remove any of the given include directive or namespace from your program.

Your work must be original. Group work is not allowed. You are required to work individually. All your work must consist of only the materials discussed in our course until the date the assignment is posted. If you use any C++ feature not discussed in the course yet, then you will not get any mark. In addition, your coding standard must be the same or similar to the coding standards discussed in the course; otherwise you will not get marks for your work. Although you don't really need to, you are free to define additional supporting or helper functions.

All your functions must have the same function name as given above including capitalization so that the given test program compiles and runs without any syntax, runtime, or semantic errors.

Starting Your Work

You are strongly recommended to copy the given test program as it is given without any modification and use it as your starting material. In order to make it easy to copy and paste the given program, a text file named **StarterCode.txt** consisting of the given test program so that you can easily copy and paste the code from this text file.

Submission Format

You will find a submission link on the course Moodle page under Week 8. You are required to submit your source code file (that is .cpp file) that must contain the given test program, the given supporting functions, and your functions definitions including any supporting function you may have defined. You must submit your work through Moodle by uploading your file using the provided submission link. No email submission is accepted for any reason.

Submission Due Date and Time

The due date and time to submit your work is Tuesday 8 November 2022 at 11:30PM. Not submitting through Moodle before the due date and time would result to zero marks. So please submit some time before the due date and time because you may encounter computer or Internet problems on the last minute and such problems will not be accepted as an excuse not to submit on time.

Marking

A program with a syntax error will get no marks at all. A program with a semantic or run time errors will be penalized depending on the severity of its shortcomings.

Sample Run Outputs

The following table gives few sample run outputs of the program once all the missing functions are implemented.

```
This program demonstrates numeric information representation using
 *** Unsigned Binary Representation
 *** Sign and Magnitude Binary Representation
 *** Two's Complement Binary Representation
```

```
In addition, the program demonstrates
 *** Two's complement binary addition, and
 *** Conversion from two's complement to decimal.

Select your computation
   1. Unsigned Binary Representation Computation
   2. Sign and Magnitude Representation Computation
   3. Two's Complement Representation Computation
   4. Exit Program
Enter your selection (1, 2, 3, or 4): 1
Enter the required bit pattern length between 1 and 32: 6
Enter a non-negative integer: 45
The unsigned binary representation of 45 in 6 bit is 101101
Press any key to continue . . .


Select your computation
   1. Unsigned Binary Representation Computation
   2. Sign and Magnitude Representation Computation
   3. Two's Complement Representation Computation
   4. Exit Program
Enter your selection (1, 2, 3, or 4): 1
Enter the required bit pattern length between 1 and 32: 4
Enter a non-negative integer: 15
The unsigned binary representation of 15 in 4 bit is 1111
Press any key to continue . . .


Select your computation
   1. Unsigned Binary Representation Computation
   2. Sign and Magnitude Representation Computation
   3. Two's Complement Representation Computation
   4. Exit Program
Enter your selection (1, 2, 3, or 4): 1
Enter the required bit pattern length between 1 and 32: 8
Enter a non-negative integer: 373
The unsigned binary representation of 373 in 8 bit is 01110101
Press any key to continue . . .


Select your computation
   1. Unsigned Binary Representation Computation
   2. Sign and Magnitude Representation Computation
   3. Two's Complement Representation Computation
   4. Exit Program
Enter your selection (1, 2, 3, or 4): 2
Enter the required bit pattern length between 1 and 32: 8
Enter an integer: 0
The sign and magnitude binary representation of 0 in 8 bit is 00000000
Press any key to continue . . .


Select your computation
   1. Unsigned Binary Representation Computation
   2. Sign and Magnitude Representation Computation
   3. Two's Complement Representation Computation
   4. Exit Program
Enter your selection (1, 2, 3, or 4): 2
Enter the required bit pattern length between 1 and 32: 5
Enter an integer: 8
The sign and magnitude binary representation of 8 in 5 bit is 01000
Press any key to continue . . .


Select your computation
```

```
   1. Unsigned Binary Representation Computation
   2. Sign and Magnitude Representation Computation
   3. Two's Complement Representation Computation
   4. Exit Program
Enter your selection (1, 2, 3, or 4): 2
Enter the required bit pattern length between 1 and 32: 5
Enter an integer: -8
The sign and magnitude binary representation of -8 in 5 bit is 11000
Press any key to continue . . .


Select your computation
   1. Unsigned Binary Representation Computation
   2. Sign and Magnitude Representation Computation
   3. Two's Complement Representation Computation
   4. Exit Program
Enter your selection (1, 2, 3, or 4): 2
Enter the required bit pattern length between 1 and 32: 8
Enter an integer: -18
The sign and magnitude binary representation of -18 in 8 bit is 10010010
Press any key to continue . . .


Select your computation
   1. Unsigned Binary Representation Computation
   2. Sign and Magnitude Representation Computation
   3. Two's Complement Representation Computation
   4. Exit Program
Enter your selection (1, 2, 3, or 4): 2
Enter the required bit pattern length between 1 and 32: 6
Enter an integer: 75
The sign and magnitude binary representation of 75 in 6 bit is 001011
Press any key to continue . . .


Select your computation
   1. Unsigned Binary Representation Computation
   2. Sign and Magnitude Representation Computation
   3. Two's Complement Representation Computation
   4. Exit Program
Enter your selection (1, 2, 3, or 4): 2
Enter the required bit pattern length between 1 and 32: 6
Enter an integer: -75
The sign and magnitude binary representation of -75 in 6 bit is 101011
Press any key to continue . . .


Select your computation
   1. Unsigned Binary Representation Computation
   2. Sign and Magnitude Representation Computation
   3. Two's Complement Representation Computation
   4. Exit Program
Enter your selection (1, 2, 3, or 4): 3
Enter the required bit pattern length between 1 and 32: 5
Enter an integer: 9
The two's complement binary representation of 9 in 5 bit is 01001
Enter a second integer: -14
The two's complement binary representation of -14 in 5 bit is 10010
The binary sum of 01001 and 10010 is 11011
The integer value of the binary sum is -5
This is a correct result.
Press any key to continue . . .


Select your computation
```

```
   1. Unsigned Binary Representation Computation
   2. Sign and Magnitude Representation Computation
   3. Two's Complement Representation Computation
   4. Exit Program
Enter your selection (1, 2, 3, or 4): 3
Enter the required bit pattern length between 1 and 32: 8
Enter an integer: 62
The two's complement binary representation of 62 in 8 bit is 00111110
Enter a second integer: -99
The two's complement binary representation of -99 in 8 bit is 10011101
The binary sum of 00111110 and 10011101 is 11011011
The integer value of the binary sum is -37
This is a correct result.
Press any key to continue . . .


Select your computation
   1. Unsigned Binary Representation Computation
   2. Sign and Magnitude Representation Computation
   3. Two's Complement Representation Computation
   4. Exit Program
Enter your selection (1, 2, 3, or 4): 3
Enter the required bit pattern length between 1 and 32: 5
Enter an integer: 11
The two's complement binary representation of 11 in 5 bit is 01011
Enter a second integer: 9
The two's complement binary representation of 9 in 5 bit is 01001
The binary sum of 01011 and 01001 is 10100
The integer value of the binary sum is -12
This is not correct result because our bit pattern is too small to store the result.
Press any key to continue . . .


Select your computation
   1. Unsigned Binary Representation Computation
   2. Sign and Magnitude Representation Computation
   3. Two's Complement Representation Computation
   4. Exit Program
Enter your selection (1, 2, 3, or 4): 3
Enter the required bit pattern length between 1 and 32: 5
Enter an integer: -321
The two's complement binary representation of -321 in 5 bit is 11111
Enter a second integer: 305
The two's complement binary representation of 305 in 5 bit is 10001
The binary sum of 11111 and 10001 is 10000
The integer value of the binary sum is -16
This is a correct result.
Press any key to continue . . .


Select your computation
   1. Unsigned Binary Representation Computation
   2. Sign and Magnitude Representation Computation
   3. Two's Complement Representation Computation
   4. Exit Program
Enter your selection (1, 2, 3, or 4): 3
Enter the required bit pattern length between 1 and 32: 3
Enter an integer: 2
The two's complement binary representation of 2 in 3 bit is 010
Enter a second integer: 2
The two's complement binary representation of 2 in 3 bit is 010
The binary sum of 010 and 010 is 100
The integer value of the binary sum is -4
This is not correct result because our bit pattern is too small to store the result.
Press any key to continue . . .
```

```
Select your computation
   1. Unsigned Binary Representation Computation
   2. Sign and Magnitude Representation Computation
   3. Two's Complement Representation Computation
   4. Exit Program
Enter your selection (1, 2, 3, or 4): 3
Enter the required bit pattern length between 1 and 32: 6
Enter an integer: 45
The two's complement binary representation of 45 in 6 bit is 101101
Enter a second integer: -17
The two's complement binary representation of -17 in 6 bit is 101111
The binary sum of 101101 and 101111 is 011100
The integer value of the binary sum is 28
This is a correct result.
Press any key to continue . . .


Select your computation
   1. Unsigned Binary Representation Computation
   2. Sign and Magnitude Representation Computation
   3. Two's Complement Representation Computation
   4. Exit Program
Enter your selection (1, 2, 3, or 4): 3
Enter the required bit pattern length between 1 and 32: 1
Enter an integer: -99
The two's complement binary representation of -99 in 1 bit is 1
Enter a second integer: 100
The two's complement binary representation of 100 in 1 bit is 0
The binary sum of 1 and 0 is 1
The integer value of the binary sum is -1
This is not correct result because our bit pattern is too small to store the result.
Press any key to continue . . .


Select your computation
   1. Unsigned Binary Representation Computation
   2. Sign and Magnitude Representation Computation
   3. Two's Complement Representation Computation
   4. Exit Program
Enter your selection (1, 2, 3, or 4): 3
Enter the required bit pattern length between 1 and 32: 1
Enter an integer: -100
The two's complement binary representation of -100 in 1 bit is 0
Enter a second integer: 99
The two's complement binary representation of 99 in 1 bit is 1
The binary sum of 0 and 1 is 1
The integer value of the binary sum is -1
This is a correct result.
Press any key to continue . . .


Select your computation
   1. Unsigned Binary Representation Computation
   2. Sign and Magnitude Representation Computation
   3. Two's Complement Representation Computation
   4. Exit Program
Enter your selection (1, 2, 3, or 4): 4
Press any key to continue . . .
```