CMPS 2433 Discrete Structures and Analysis
Program 1 Fall 2022

Two Part Problem:

A. Write a function ***`createBitwiseTable(bitTable1`***) to compute & return the resulting values of the following bitwise operations. If the operation is a binary operation, use the first column X for the first operand and the second column Y for the second operand. If it is a unary operation, then perform the operation on the first column X. The bitwise operations to compute are &, |, ^, ~.
Basically create a truth table and then add rows for more bitwise ops on larger numbers.

   Write a second function to populate table 1's first and second columns with values – hard core the basic true table and then input N values from the file.

   Write a third function to print the resulting table 1.

B. Write a function ***`moreBitwiseOps(bitTable2)`*** to compute & return the resulting values of the following bitwise operations. Again use the first column X for the first operand and the second column Y for the second operand. If it is a unary operation, then perform the operation on the first column X. The operations to compute are setbit(i), getbit(i) shift left, shift right, isEven, countOnes and isPower2.

   Write a second function to initialize table 2 to -1s.

   Write a third function to print the resulting table 2.

   Hint for isPower2 O(1) algorithm:

   Consider a number x that we need to check for being a power for 2. Now think about the binary representation of (x-1). (x-1) will have all the bits same as x, except for the rightmost 1 in x and all the bits to the right of the rightmost 1.
   Le, x = 4 = $(100)_2$, then x - 1 = 3 = $(011)_2$
   Le, x = 6 = $(110)_2$, then x - 1 = 5 = $(101)_2$

   It might not seem obvious with these examples, but the binary representation of (x-1) can be obtained by simply flipping all the bits to the right of rightmost 1 in x and also including the rightmost 1.

   Now think about x & (x-1). x & (x-1) will have all the bits equal to the x except for the rightmost 1 in x.

   Let, x = 4 = $(100)_{2,}$ then x - 1 = 3 = $(011)_2$ and x & (x-1) = 4 & 3 = $(100)_2$ & $(011)_2$ = $(000)_2$
   Let, x = 6 = $(110)_2$, then x - 1 = 5 = $(101)_2$ and x & (x-1) = 6 & 5 = $(110)_2$ & $(101)_2$ = $(100)_2$

   Properties for numbers that are powers of 2, is that they have one and only one bit set in their binary representation. If the number is neither zero nor a power of two, it will have 1 in more than one place. So if x is a power of 2 then x & (x-1) will be 0.

Input:

Input will be from the file named **"program1.dat".** The first line of input gives the number of rows of data, *N* (at most 25) for problem A beyond the basic truth table. These N lines either 2 operands. Then the next line with have the number of rows of data, *M* (at most 25) for problem B. The next M lines will have the operator (a char) and either 1 or 2 or 3 operands.

The char operator *should* be self-explanatory for the op that needs to be performed – use a switch statement inside a for loop.

Create another input file program1B.data and put different data in it to run the program 2nd time.

Output:

Two output files. Each output file has your name, program name, and two tables with all the input in columns X and/or Y (or P and B) and the operations across the columns. If a column is empty, print a blank instead.

Turn in:

- Printout of Source code file (.cpp)
- Both input files
- Both output files
- Upload .cpp file to D2L

Sample input:

```
3
2 0
0 2
35 7
9
> 35
< 35
S 35 3 1
E 34
G 35 3
2 0
C 35
2 2
2 48
```

Corresponding Output

Nora Jones
Program 1: Bitwise Operations

X	Y	&	|	^	~
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0
2	0	0	2	2	65533
0	2	0	2	2	65535
35	7	3	39	36	65500

X	P	V	>>	<<	S	G	E	C	2
35			17						
35				70					
35	3	1			43				
34							1		
35	3					0			
0									0
35								3	
2									1
48									0