UMass Boston CS 240
Homework 4
Due 11/04/2019 19:00

This assignment is to use bitwise operations to add two numbers.
No late homework is accepted.

1 Half Adder and Full Adder

In digital circuits, the half adder adds two input bits, P and Q, and produces two output bits, sum S and carry C. See the left part of Figure 1. The truth table of the half adder is as follows.

input		output	
P	Q	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Ideally, we should write one function that computes both S and C, but we have not discussed how to return multiple values from a function. Therefore, we write two functions that compute S and C separately, as follows.

```
enum bits {ZERO, ONE};

enum bits halfAdderSum(enum bits P, enum bits Q)
```

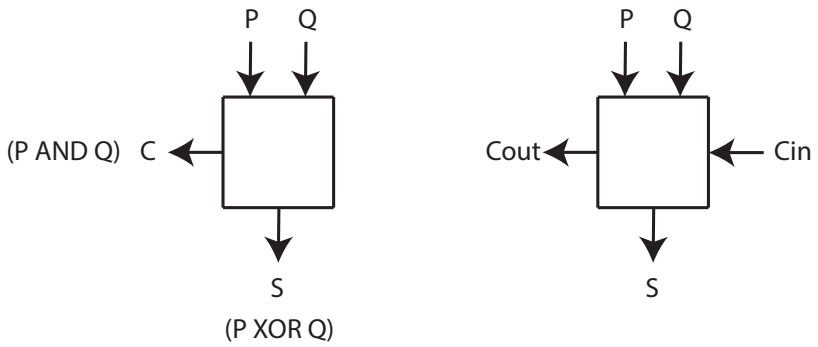

Figure 1: Left: the half adder. Right: the full adder

```
{
  return P ^ Q;
}

enum bits halfAdderCarry(enum bits P, enum bits Q)
{
  return P & Q;
}
```

The half adder adds only two bits. It becomes inadequate when we want to add more than two bits. Let us consider adding two 8-bit integers, $P_7P_6P_5P_4P_3P_2P_1P_0$ and $Q_7Q_6Q_5Q_4Q_3Q_2Q_1Q_0$. After we add P_0 and Q_0, the carry bit may be 1. We must incorporate it when adding P_1 and Q_1. Thus we need the full adder that takes three inputs: P, Q, and the carry-in Cin. The output bits are sum S and carry-out Cout. See the right part of Figure 1. The truth table of the full adder is as follows.

input			output	
P	Q	Cin	Cout	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Your task in this part of the homework is to implement the two functions for the full adder in the file `adder.c`.

```
enum bits {ZERO, ONE};

enum bits fullAdderSum(enum bits P, enum bits Q, enum bits Cin)
{
}

enum bits fullAdderCarry(enum bits P, enum bits Q, enum bits Cin)
{
}
```

You should start by working out the Boolean expressions for S and Cout.

2 Adding Two Numbers

Now you are ready to add two 32-bit integers bit by bit. Using a cascade of full adders, you can add a pair of bits and the carry bit from the previous position, save the sum bit, and send the carry bit to the next position. The first Cin should be set to zero. If we assume the sum of the two numbers does not overflow the 32-bit storage, we can safely discard the last Cout.

Your task in this part of the homework is to implement `myAdd()` in the file `myAdd.c`. The algorithm works as follows.

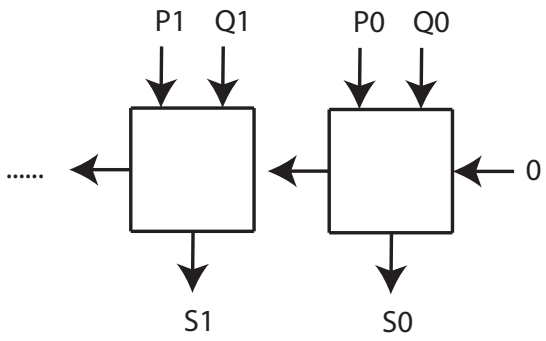

Figure 2: Using a cascade of full adders to add two numbers.

1. Initialize Cin to zero
2. For $i = 0, 1, \ldots, 31$
   (a) Extract the i-th bits from P and Q
   (b) Use `fullAdderSum()` and `fullAdderCarry()` to calculate the sum bit S and the carry bit Cout
   (c) Write the sum bit to the i-bit of `mySum`
   (d) Move Cout to Cin for the next iteration

There are many ways to extract a bit or write a bit in a 32-bit storage. I will suggest one way here:

```
    For an integer i,
    (i & 1 << 0 )!= 0
    (i & 1 << 1 )!= 0
    (i & 1 << 2 )!= 0
these three operations will get the corresponding first, second, and third bit
of integer i.

    For an integer i which is internalized to 0 and bits b,
    (bits here is the above enum type)
    i = i | b << 0
    i = i | b << 1
    i = i | b << 2
these three statements will write the bits of b into the corresponding first,
second, and third bit of integer i.

Actually you can also write a branch here, because if b is 0, there is no need
to do everything. But as long as it works.
```

3 The Driver

The driver is in the file `main.c`. You should not change anything in the main program.

4 Grading

Inside hw4: adder.h, adder.c, main.c, myAdd.c