Problem 1: The routine samples in the code below is called many times. Consider the execution of the code on three systems, each system using one of the branch predictors below.

All predictors use a $2^{14}$-entry branch history table (BHT). (The global predictor does not need its BHT for predicting branch direction.) The three predictors are:

- System B: bimodal
- System G: global, history length 10. (Accuracy can be approximated.)
- System L: local, history length 10.

(a) Determine the amount of memory (in bits) needed to implement each predictor.

(b) For each loop in samples determine the accuracy of the loop branch (the one that tests the value of i) after warmup on each system. The accuracy for the global predictor can be approximated, the others must be determined exactly.

(c) Why would solving the problem above be impossible, or at least tedious, if the BHT size were $\approx 2^3$ entries?

```c
void samples(int& x, int& y, char **string_array )
{
    // Loop 5-xor
    for( int i = 0; i < 5; i++ )
        x = x ^ i;

    // Loop 5-len
    for( int i = 0; i < 5; i++ )
        if( strlen( string_array[i] ) < 20 )
            return; // Never executes. <- Important.

    // Loop 100-xor
    for( int i = 0; i < 100; i++ )
        y = y ^ i;
}
```

There’s more on the next page.
Problem 2: The code `more`, below, runs on four systems. All predictors use a $2^{14}$-entry branch history table (BHT). (The global and gshare predictors do not need its BHT for predicting branch direction.) The predictors are:

- System B: bimodal
- System G: global, history length 10. (Accuracy can be approximated.)
- System X: gshare, history length 10. (Accuracy can be approximated.)
- System L: local, history length 10.

(a) In the code below estimate the prediction accuracy of the following predictors on Branch B and Branch C (there is no Branch A) after warmup, assuming that `more` is called many times.

(b) One of the predictors should have a low prediction accuracy. Why? Avoid a sterile description of the hardware, instead discuss the concept the predictor is based on and why that’s not working here.

```c
void more(int& x, int& y, int a, int& b, int& c)
{
    for( int i=0; i<100; i++ ) x = x ^ i;

    if( a < 10 ) b++; // Branch B, never taken.
    for( int i=0; i<100; i++ ) y = y ^ i;

    if( a >= 10 ) c++; // Branch C, always taken.
}
```