For the following problems, consider the following code once it reaches steady-state:

```
do {
    for (int i = 0; i < 4; i++) {
        // increment something
    }
    for (int j = 0; j < 8; j++) {
        // increment something
    }
    k++;
} while (k < 1000000000)
```

The corresponding assembly is something close to this:

```
    MOV $t3, 4          # t3=4
    MOV $t4, 8          # t4=8
    MOV $t5, 1000000000 # t5=1000000000
    MOV $t0, 0          # k=0
A:  MOV $t1, 0          # i=0
B:  # increment something
    ADDI $t1, $t1, 1    # i+=1
    BLT  $t1, $t3, B    # if i<4, goto B
C:  MOV  $t1, 0         # j=0
D:  # increment something
    ADDI $t1, $t1, 1    # i+=1
    BLT  $t1, $t4, D    # if j<8, goto D
    BLT  $t0, $t5, A    # if k<1000000000, goto A
```

1. **Static Branch Prediction** (1 point each)

   (a) What is the branch prediction accuracy for an always not-taken (`PC+4` prediction) branch predictor?

   2/13 = 15% because all branches predicted not-taken, but only two branches (at the end of each for loop) are not taken.

   (b) What is the branch prediction accuracy for an always taken branch predictor?

   11/13 = 85% because all branches predicted taken, but two branches (at the end of each for loop) are not taken.

2. **Dynamic Branch Prediction** (2 points each)

   (a) What is the branch prediction accuracy for a 1-bit branch predictor?

9/13 = 69%. Both L8 and L12 are mispredicted on the first and last time. L13 is always mispredicted.

(b) What is the branch prediction accuracy for a 2-bit branch predictor?

11/13 = 84.6%. L12 is no longer mispredicted the first time (only the last). L13 is no longer mispredicted.

3. **Global vs. Local Branch Prediction** (3 points each)

Assume that the PHT contains 2-bit counters.

(a) What is the branch prediction accuracy for a global branch predictor with a 5-bit history?

	test	value	GR	results
1	`i<4`	`i=0`	`11101`	taken
2	`i<4`	`i=1`	`11011`	taken
3	`i<4`	`i=2`	`10111`	taken
4	`i<4`	`i=3`	`01111`	taken
5	`i<4`	`i=4`	`11111`	**not taken**
6	`j<8`	`j=0`	`11110`	taken
7	`j<8`	`j=1`	`11101`	taken
8	`j<8`	`j=2`	`11011`	taken
9	`j<8`	`j=3`	`10111`	taken
10	`j<8`	`j=4`	`01111`	taken
11	`j<8`	`j=5`	`11111`	**taken**
12	`j<8`	`j=6`	`11111`	**taken**
13	`j<8`	`j=7`	`11111`	**taken**
14	`j<8`	`j=8`	`11111`	**not taken**
15	`k<1000000000`	`k=?`	`11110`	taken

11/13 = 84.6%.

We only really have to look at the conflicting one: `11111`. With either a 1 or two bit predictor, both 14 and 11 will be mispredicted because there have been at least two "proofs" in the opposite direction.

(b) What is the branch prediction accuracy for a local branch predictor with a 5-bit history?

12/13 = 92.3%. 5 is no longer mispredicted because it is a separate branch. 14 is still mispredicted, but because there aren't two not-taken's is a row, the predictor will be on "weakly taken" when 11 is run, so it will predict correctly.