Consider the following C code:

```c
int i = 0, sum_remainder_0 = 0, sum_remainder_1 = 0, sum_remainder_2 = 0;
do {
    if(i % 3 == 0) // Branch X: taken if i % 3 == 0
        sum_remainder_0 += A[i];
    else
    {
        if(i%3 == 1) // Branch Y: taken if i % 3 == 1
            sum_remainder_1 += A[i];
        else
            sum_remainder_2 += A[i];
    }
    i ++;
} while (i < 100); // Branch Z: taken if i < 100
```

Please answer the following questions:

A. If your processor uses a “always-taken” predictor, what’s the branch prediction accuracy?
B. If your processor uses a “global 2-bit” predictor with 4-bit GHR and unlimited BTB entries, what’s the branch prediction accuracy? Assume all counters are initialized as 0s.
2. For the following C code, where elements within the same row are stored contiguously, assume each word is a 32-bit integer:

```c
for(i = 0; i < 8; i++)
    for(j = 0; j < 8000; j++)
        a[i][j] = b[i][0] + a[j][i]
```

A. How many 32-bit integers can be stored in a 16-byte cache block?

B. References to which variable exhibit temporal locality?

C. References to which variable exhibit spatial locality?
3. For the following memory access sequence in decimal numbers, given as “byte-addresses”

3, 180, 43, 2, 191, 88, 190, 14, 181, 44, 186, 253

A. Please rewrite this address sequence in binary.

B. Assume the cache only contains a total of one 16-byte block, please identify the tag field of each address in the access sequence and list if each reference is a hit or miss.