Problem 1:
Convert the 2-digit two's complement hexadecimal integer 0xb5 to decimal. (20 points)

ANSWER: -75

0xb5 = 1011 0101 = -128 + 32 + 16 + 4 + 1 = -75

Problem 2:
Convert the decimal integer 54 to an 8-bit two's complement binary integer. (20 points)

ANSWER: 0011 0110

54 = 32 + 16 + 6 = 0011 0110

Problem 3:
You're given two 4-digit, 2's complement hexadecimal numbers X = 0x9237 and Y = 0xf5c4. Compute X - Y. Remember to indicate overflow if it occurs. Show all intermediate steps clearly. (30 points)

ANSWER: 0x9c73

-Y = ffff - f5c4 + 1 = 0a3c

X - Y = 9237 + 0a3c = 9c73 (no overflow)
4) Circle a language: C++ or Java

Write the function/method findKeys(), with three arguments:

- data[] an array of integers (input)
- key an integer; the number to search for (input)
- size an integer; number of elements in data[] (input)
- k[] an array of integers (output)

k[] starts out with zero elements.

(For Java programmers, you can assume that data[] and k[] have been allocated more space than you need, i.e., data.length and k.length are very large.)

findKeys() searches for elements of data[] that match the integer key, copies the index of each matching element in data[] to k[], in order, and returns the number of matching elements.

For example, if data[] = {1, 2, 2, -2, 2}, key = 2, and size = 5, when findZeros() returns, k[] will contain {1, 2, 4}, and a 3 will be returned.

Write the code for findKeys(). Show the prototype for findKeys() clearly. You don’t have to write a main program.

30 points

[Java version:]

```java
int findKeys(int data[], int key, int size, int k[]) {
    int count = 0;

    for (int i=0; i<size; i++) {
        if (data[i] == key) {
            k[count] = i;
            count++;
        }
    }
    return count;
}
```