Problem 1:
Convert the decimal integer 82 to 2-digit two’s complement hexadecimal. (20 points)

ANSWER: 0x52

82 = 64 + 16 + 2 = 0101 0010 = 0x52

Problem 2:
Convert the 8-bit two’s complement binary integer 1100 0101 to decimal. (20 points)

ANSWER: -59

1100 0101 = -128 + 64 + 5 = -59

Problem 3:
You’re given two 4-digit, 2’s complement hexadecimal numbers X = 0x32c2 and Y = 0x20f4. Compute X-Y. Remember to indicate overflow if it occurs. Show all intermediate steps clearly. (30 points)

ANSWER: 0x11ce

-Y = ffff – 20f4 + 1 = 0xdf0c
X + (-Y) = 0x32c2 + 0xdf0c = 0x11ce
4) Circle a language: C++ or Java

Write a function/method in that language with the following prototype/header:

    int insert(int x[], int num);

x[] is an array of non-negative integers; the end of the array is marked with a negative integer. When the method is called, x[] is already sorted in non-descending order. The method inserts the array element num into x[], while maintaining the sorted order of x[], and marking the end of the array with a negative integer. The method then returns the index of the position that num was stored into. You may assume that the number of elements in x[] is big enough for the insertion to work correctly.

(For Java programmers, x[] is allocated x.length elements, but some of the elements are unused. The negative sentinel value marks the end of the valid elements in the array.)

For example:

    int arr[] = {2, 7, 10, -1, 4}; // arr[] has 3 elements, 2, 7, 10, sorted
    int y;

    int y = insert(arr, 9);


ANSWER:

    int insert(int x[], int num) {
        int count = 0;
        int i=0;
        while (x[i] < num && x[i] >= 0) {
            i++;
        }
        count = i;
        int j;
        for (j=count; x[j] >= 0; j++);

        for (int k=j+1; k>count; k--)
            x[k] = x[k-1];
        x[count] = num;
        return count;
    }