**8. Pandora’s Box**

# Program Name: PandorasBox.java Input File: pandorasbox.dat

Pandora has created a special type of 4 digit lock to protect her box from being opened. It will always display a 4 digit value, it also has a special unlock code (integer value). The lock will only open when the unlock code is displayed. To help unlock it, there are some special buttons available with that lock, each with a numerical value. Whenever the button is pressed, its value is added to the value of the display, thus a new number is displayed. The locks always uses least significant four digits after addition. Being the sneaky person you are, you have decided to make a program to break the code on the lock.

**Input**

For each test case, there will be 3 numbers associated with it: X, Y and Z where X (0000 ≤ X ≤ 9999) represents the current code on the display, Y (0000 ≤ Y ≤ 9999) represents the unlock code and Z (1 ≤ Z ≤ 10) represents the number of buttons on that lock. The next Z numbers (0 ≤ V ≤ 9999) are in a single line after that, separated by a single white space, and they represent the values of the buttons themselves. The values of X, Y, V will always be denoted by a four digit number, even if padding of leading zeroes is used. The input file will end when X = Y = Z = 0.

**Output**

For each test case, print the word Case followed by the test case number and a : , a space and by the minimum number of button presses required to unlock Pandora’s lock. The numbers do not “roll over” like a traditional padlock, i.e. 9 does not turn into 0. You can only add or subtract values but not both. If no solution exists, then print Cannot be Opened in place of the minimum number of presses required.

**Example Input File**

5234 1212 3

1023 0101 0001

0000 9999 1

1000

0000 9999 1

0001

0 0 0

**Example Output to Screen**

Case 1: 48

Case 2: Cannot be Opened

Case 3: 9999