CISS240: Introduction to Programming
Assignment 9

Name: ____

Write a program that prompts the user for 4 integers values for a, b, c, d and prints a table of c–powers to d–powers for integers from a to b. In the output below, the column widths are all set to 15.

Test 1.

```
3 10 4 7
                      4              5              6              7
        ------------------------------------------------------------
     3|              81            243            729           2187
     4|             256           1024           4096          16384
     5|             625           3125          15625          78125
     6|            1296           7776          46656         279936
     7|            2401          16807         117649         823543
     8|            4096          32768         262144        2097152
     9|            6561          59049         531441        4782969
    10|           10000         100000        1000000       10000000
```

Test 2.

```
-5 5 4 8
                      4              5              6              7              8
        ---------------------------------------------------------------------------
    -5|             625          -3125          15625         -78125         390625
    -4|             256          -1024           4096         -16384          65536
    -3|              81           -243            729          -2187           6561
    -2|              16            -32             64           -128            256
    -1|               1             -1              1             -1              1
     0|               0              0              0              0              0
     1|               1              1              1              1              1
     2|              16             32             64            128            256
     3|              81            243            729           2187           6561
     4|             256           1024           4096          16384          65536
     5|             625           3125          15625          78125         390625
```

Test 3.

```
1 10 2 2
                    2
      ---------------
    1|              1
    2|              4
    3|              9
    4|             16
    5|             25
    6|             36
    7|             49
    8|             64
    9|             81
   10|            100
```

Test 4.

```
20 20 2 5
                    2              3              4              5
      ------------------------------------------------------------
   20|            400           8000         160000        3200000
```

Write the following program that prints a calendar month. The program prompts the user for three integers: the month, the year, and the day–of–week for the first day of the month (with 0 representing Sunday, 1 representing Monday, etc.) For instance, if the user entered

```
3 2008 6
```

It means that he/she wants the calendar for March 2008 and the first day of the month is a Saturday. The output is

```
March 2008
--------------------
Su Mo Tu We Th Fr Sa
                   1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31
```

See the "Hints"section for hints and suggestions.

Test 1.

```
3 2008 6
March 2008
--------------------
Su Mo Tu We Th Fr Sa
                   1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31
```

Test 2.

```
3 2008 4
March 2008
--------------------
Su Mo Tu We Th Fr Sa
             1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30 31
```

Test 3.

```
2 2008 5
February 2008
--------------------
Su Mo Tu We Th Fr Sa
                1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29
```

Test 4.

```
2 2009 0
February 2009
--------------------
Su Mo Tu We Th Fr Sa
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
```

Of course your program should be smart enough to compute the number of days in the month (including leap year cases for the month of February.)

Spoiler Warning . . . Incoming Hints . . .

Analyze the problem carefully and try to break it down to smaller sub–problems. Look at **Test 1:**

```
3 2008 6
March 2008
--------------------
Su Mo Tu We Th Fr Sa
                   1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31
```

STEP 1. Try to get your program just to do this:

```
3 2008 6
March 2008
--------------------
Su Mo Tu We Th Fr Sa
```

This should be easy. Make sure you try different test cases.

STEP 2. When that's done, you can try to get your program to do this:

```
3 2008 6
March 2008
--------------------
Su Mo Tu We Th Fr Sa
                   1
```

In other words try to print the first day of the month at the right spot. Again, try as many test cases as possible, making sure that the first day of the month is always printed at the right place.

STEP 3. Next print all the days in the month without wrap–around.

```
4 2008 2
April 2008
--------------------
Su Mo Tu We Th Fr Sa
       1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
```

STEP 4. Next get your program to print newline at the right places.

```
4 2008 2
April 2008
--------------------
Su Mo Tu We Th Fr Sa
       1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30
```

Brute-force search for integer solutions in mod n

Find all solutions to this problem:

$$x * x \,\%\, n = a \,\%\, n$$
$$x \text{ is an integer between } 0 \text{ and } n-1$$

Note that this means that we are only interested in remainders mod n (that's why we are only interested in $0, 1, 2, \ldots, n-1$.) Using the correct mathematical notation in Computer Science (especially in the area of Cryptography) one would say that we're solving this equation:

$$x * x \equiv a \pmod{n}$$

or

$$x^2 \equiv a \pmod{n}$$

The program prompts the user for n and a, prints all the solutions, and prints the number of solutions.

Let's do one by hand. Let's try $n = 5$ and $a = 1$. Then, the equation we are looking at is this:

$$(x * x) \,\%\, 5 = 1 \,\%\, 5$$
$$x \text{ is an integer between } 0 \text{ and } 4$$

i.e.,

$$(x * x) \,\%\, 5 = 1$$
$$x \text{ is an integer between } 0 \text{ and } 4$$

Next, we try all the integers from 0 to 4 and see which one is a solution:

x	$(x * x) \,\%\, 5$
0	$(0 * 0) \,\%\, 5 = 0 \,\%\, 5 = 0$
1	$(1 * 1) \,\%\, 5 = 1 \,\%\, 5 = 1$
2	$(2 * 2) \,\%\, 5 = 4 \,\%\, 5 = 4$
3	$(3 * 3) \,\%\, 5 = 9 \,\%\, 5 = 4$
4	$(4 * 4) \,\%\, 5 = 16 \,\%\, 5 = 1$

The solutions are $x = 1, x = 4$. (See Test 1.)

Test 1.

```
5 1
1 4
2
```

Test 2.

```
19 2

0
```

Test 3.

```
19 4
2 17
2
```

Test 4.

```
19 5
9 10
2
```

(Note: Is it true that there are always either none or exactly two solutions when n is prime?)

Brute force search for integer solutions

Here's something from Wikipedia:

> 1729 is known as the Hardy-Ramanujan number, after a famous anecdote of the British mathematician G. H. Hardy regarding a hospital visit to the Indian mathematician Srinivasa Ramanujan. In Hardy's words:
>
> "I remember once going to see him when he was ill at Putney. I had ridden in taxi cab number 1729 and remarked that the number seemed to me rather a dull one, and that I hoped it was not an unfavorable omen. "No,"he replied,"it is a very interesting number; it is the smallest number expressible as the sum of two cubes in two different ways.""
>
> Numbers such as
>
> $$1729 = 1^3 + 12^3 = 9^3 + 10^3$$
>
> that are the smallest number that can be expressed as the sum of two cubes in n distinct ways have been dubbed taxicab numbers. 1729 is the second taxicab number (the first is $2 = 1^3+1^3$). The number was also found in one of Ramanujan's notebooks dated years before the incident.

Write a program that prompts the user for an integer z and finds all positive (i.e., at least 1) integer solutions to the equation

$$x^3 + y^3 = z$$

Do not list repeats. For instance when z is 1729

$$1^3 + 12^3 = 1729$$
$$12^3 + 1^3 = 1729$$

But only $1, 12$ should be listed (i.e., do not list $12, 1$.)

[Hint: Write a double for-loop.]

Test 1.

```
1
```

Test 2.

```
2
1,1
```

Test 3.

```
1729
1,12 9,10
```

Test 4.

```
2000
10,10
```

The above exercise verified that there are exactly two integer solutions. It does not verify that 1729 is the "**smallest number expressible as the sum of two cubes in two different ways**". To do that you need to solve this:

$$x^3 + y^3 = z$$

where z runs from 1 to 1729.

Write a program that prompts the user for an integer variable n and prints solutions to

$$x^3 + y^3 = z$$

for positive integers x, y, z where z runs from 1 to n. (Here, positive integers means integers which are at least 1.) You should print out x, y, and z only when there is at least one solution. See the output for the format.

Test 1.

```
1729
2: 1,1
9: 1,2
16: 2,2
28: 1,3
35: 2,3
54: 3,3
65: 1,4
72: 2,4
91: 3,4
126: 1,5
128: 4,4
133: 2,5
152: 3,5
189: 4,5
217: 1,6
224: 2,6
243: 3,6
250: 5,5
280: 4,6
341: 5,6
344: 1,7
351: 2,7
370: 3,7
407: 4,7
432: 6,6
```

```
468: 5,7
513: 1,8
520: 2,8
539: 3,8
559: 6,7
576: 4,8
637: 5,8
686: 7,7
728: 6,8
730: 1,9
737: 2,9
756: 3,9
793: 4,9
854: 5,9
855: 7,8
945: 6,9
1001: 1,10
1008: 2,10
1024: 8,8
1027: 3,10
1064: 4,10
1072: 7,9
1125: 5,10
1216: 6,10
1241: 8,9
1332: 1,11
1339: 2,11
1343: 7,10
1358: 3,11
1395: 4,11
1456: 5,11
1458: 9,9
1512: 8,10
1547: 6,11
1674: 7,11
1729: 1,12 9,10
```

Test 2.
The input for the following test is 5000. I'm only showing part of the output because the full listing is too long.

```
5000
... OUTPUT NOT DISPLAYED HERE ...
4075: 11,14
4097: 1,16
4104: 2,16 9,15
4123: 3,16
4160: 4,16
4221: 5,16
4312: 6,16
4375: 10,15
4394: 13,13
4439: 7,16
4472: 12,14
4608: 8,16
4706: 11,15
4825: 9,16
4914: 1,17
4921: 2,17
4940: 3,17
4941: 13,14
4977: 4,17
```

Write a program that prompts the user and produces the following ASCII art.

Test 1.

```
1
*
```

Test 2.

```
2
* *
 *
```

Test 3.

```
3
*   *
 * *
  *
```

Test 4.

```
4
*     *
 *   *
  * *
   *
```

Another crazy pesky ASCII art problem . . .

Test 1.

```
1
*
```

Test 2.

```
2
 *
***
```

Test 3.

```
3
  *
 ***
*****
```

Test 4.

```
4
  *
 ***
*****
 ***
```

Test 5.

```
5
  *
 ***
*****
 ***
*****
```

Test 6.

```
6
   *
  ***
 *****
  ***
 *****
*******
```

Test 7.

```
7
   *
  ***
 *****
  ***
 *****
*******
 *****
```

Test 8.

```
8
   *
  ***
 *****
  ***
 *****
*******
 *****
*******
```

Test 9.

```
9
    *
   ***
  *****
   ***
  *****
 *******
  *****
 *******
*********
```

Test 10.

```
10
    *
   ***
  *****
   ***
  *****
 *******
  *****
 *******
*********
 *******
```

Test 11.

```
11
    *
   ***
  *****
   ***
  *****
 *******
  *****
 *******
*********
 *******
*********
```

Test 12.

```
12
     *
    ***
   *****
    ***
   *****
  *******
   *****
  *******
 *********
  *******
 *********
***********
```

Spoiler Warning . . . Incoming Hints . . .

This problem is not as difficult as you think. This is Test 10:

Test 10

```
10
    *
   ***
  *****
   ***
  *****
 *******
  *****
 *******
*********
 *******
```

Of course, you can see that there are ten lines. You would expect a for-loop where `i` runs from 1 to 10. Each iteration of the loop basically prints some spaces, some stars, and a newline. The question is this: how many spaces and how many stars?

Each time you go to the next line, the number of spaces and number of stars changes in a certain way.

The general structure of the code for the case when the user enters 10 is this:

```
prompt user for n
numSpaces = ???
numStars = ???

for i = 1, 2, 3, 4, ..., n:
    print numSpaces spaces
    print numStars stars
    print newline (i.e., std::endl)

    modify numSpaces
    modify numStars
```

I suggest you initialize `numSpaces` to 20 and try to get `numStars` correct first:

```
prompt user for n
numSpaces = 20 // temporary!!!
numStars = ???
```

```
for i = 1, 2, 3, 4, ..., n:
    print numSpaces spaces
    print numStars stars
    print newline (i.e. std::endl)

    update numStars
    update numSpaces
```

Complete this table and see the pattern when you run the program with $n = 10$.

```
i = 1    numStars = 1
i = 2    numStars = 3
i = 3    numStars = 5
i = 4    numStars = 3
```

Here's a final hint: `numStars` goes up by 2 when a condition is true and goes down by 2 when that condition is false. (This tells you that this is an if-else). What is that condition? (Of course the condition depends on `i`.)

In fact, the condition is used for the update on `numSpaces` as well. In other words, the code looks like this:

```
prompt user for n
numSpaces = 20 // temporary
numStars = ???
for i = 1, 2, 3, 4, ..., n:
    print numSpaces spaces
    print numStars stars
    print newline (i.e., std::endl)

    if (_______________):
        increment numStars by 2
        decrement numSpaces by ___
    else:
        decrement numStars by 2
        increment numSpaces by ___
```

Back to the initialization of `numSpaces`. There are many ways of doing this. Look at Test 10:

Test 10

```
10
    *
   ***
  *****
   ***
  *****
 *******
  *****
 *******
*********
 *******
```

The first line has 4 spaces. Why 4 spaces? Look at the line of output with the maximum number of stars. See the 9 stars?

Test 10:

```
10
1234*
   ***
  *****
   ***
  *****
 *******
  *****
 *******
*********
 *******
```

[OPTIONAL]

ASCII art strikes again!

Test 1.

```
3
***
* *
***
```

Test 2.

```
5
*****
*   *
* * *
*   *
*****
```

Test 3.

```
7
*******
*     *
* *** *
* * * *
* *** *
*     *
*******
```

Test 4.

```
9
*********
*       *
* ***** *
* *   * *
* * * * *
* *   * *
* ***** *
*       *
*********
```

[OPTIONAL]

Uh-oh . . . another ASCII art program.

Test 1.

```
1
  *
 ***
*****
```

Test 2.

```
2
     *
    ***
   *****
  *     *
 ***   ***
***** *****
```

Test 3.

```
3
        *
       ***
      *****
     *     *
    ***   ***
   ***** *****
  *     *     *
 ***   ***   ***
***** ***** *****
```

Test 4.

```
4
           *
          ***
         *****
        *     *
       ***   ***
      ***** *****
     *     *     *
    ***   ***   ***
   ***** ***** *****
  *     *     *     *
 ***   ***   ***   ***
***** ***** ***** *****
```

Spoiler Warning . . . Incoming Hints . . .

Look at the case of Test 4:

Test 4

```
4
           *
          ***
         *****
        *     *
       ***   ***
      ***** *****
     *     *     *
    ***   ***   ***
   ***** ***** *****
  *     *     *     *
 ***   ***   ***   ***
***** ***** ***** *****
```

There are two possible outermost for-loop:

Either

```
        for i = 1, 2, ..., 12
            [some code to draw 1 line]
```

or

```
        for i = 1, 2, 3, 4:
            [some code to draw 3 lines]
```