# Hints for Lab 4



## Task A. Box

Write a program `box.cpp` that asks the user to input `width` and `height` and prints a solid rectangular box of the requested size using asterisks.

Also, print a line `Shape:` between user input and the printed shape (to separate input from output).

**Example:**

```
Input width: 7
Input height: 4

Shape:
*******
*******
*******
*******
```

Hints for Task A

(4) To **do something** for x times, where x is an int. Use the following pattern.

```
for (int i = 0; i < x; i++)
    do something;
```

For example, to print out 7 asterisks, use the following statement.

```
for (int i = 0; i < 7; i++)
    cout << "*";
```

To print out width asterisks, use the following statement.

```
for (int i = 0; i < width; i++)
    cout << "*";
```

(4) In each row, we do two things:
(2.1) print out * for width times.
(2.2) print out a new line character.

(4) Do things in (2) for height rows. Since there are more than one statement in that loop body, enclose these statements in a pair of curly braces { and }.

(4) Do not forget to print “Shapes:” in a separated line before nested loop for purpose of automatic grading.

## Task B. Checkerboard

Write a program `checkerboard.cpp` that asks the user to input `width` and `height` and prints a rectangular checkerboard of the requested size using asterisks and spaces (alternating).

**Example:**

```
Input width: 11
Input height: 6

Shape:
* * * * * *
 * * * * *
* * * * * *
 * * * * *
* * * * * *
 * * * * *
```

**Hint:**

You used nested loops in the previous task that looked probably like

```
for (int row = 0; row < height; row++) {
    for (int col = 0; col < width; col++) {

        ...
```

```
    }
}
```

Inside the loops, you can add an `if` statement that will be conditionally printing asterisk `*` or `(space)` depending on the coordinates `row` and `col`. That is,

(1) When row is even (ie, the remainder of row divided by 2 is 0. Remainder operator, aka modular operator, is written as % in C++) and col is even, what character to print?
(2) When row is even and col is odd, what character to print?
(3) When row is odd and col is even, what character to print?
(4) When row is odd and col is odd, what character to print?

Another way of thinking is, printing asterisk or space depends the parity of row and col at a location is the same or not. That is, if the parity of row and col are the same, what to print, otherwise, what to print. If you use this way of thinking, you can work on Task G: checkerboard (3x3) easily.

## Task C. Cross

Write a program `cross.cpp` that asks the user to input the shape `size`, and prints a diagonal cross of that dimension.

**Example:**

```
Input size: 8

Shape:
*       *
 *     *
  *   *
   **
   **
  *   *
 *     *
*       *
```

Hint:

1. Draw a cross with size 4.

```
*  *
 **
 **
*  *
```

2. Draw a cross with size 5.

```
*   *
 * *
  *
 * *
*   *
```

3. When size is even, divide the shape by top half and bottom half. When size is odd, divide the shape by top half, middle part (with only one asterisk) and bottom half.

4. For each half, for each row, observe the only differences are preceding spaces before the first asterisk and spaces between the two asterisks. Next, find out the number of preceding spaces (those before the leftmost asterisk), number of middle spaces (those between the two asterisks). Here is a pseudocode.

   (1) enter size

   (2) print “Shape: “ in a new line.

   (3) //The following code draws the upper half.

   for each row

   begin

   (3.1) Draw spaces for number of preceding spaces times.

   (3.2) Draw the first asterisk.

   (3.3) Draw spaces for number of mid spaces times.

   (3.4) Draw the second asterisk.

   (3.5) Draw a new line.

   (3.5) Update number of preceding spaces for the next row.

   (3.6) Update number of mid spaces for the next row.

   end

(4) If size is odd, print * to the middle point (you need to find out number of preceding spaces before that *)

(5) Draw the bottom half similar to the procedure of the top half, listed in Step (3).

## Task D. Lower triangle

Write a program `lower.cpp` that prints the bottom-left half of a square, given the side `length`.

**Example:**

```
Input side length: 6

Shape:
*
**
***
****
*****
******
```

Hint:

Label the first row by 1, Increase row number by 1 when we move to the next row. In each row, print as many as asterisks as the label of row.

## Task E. Upper triangle

Write a program `upper.cpp` that prints the top-right half of a square, given the side `length`.

**Example:**

```
Input side length: 5

Shape:
*****
 ****
  ***
```

```
   **
    *
```

Hint:
For each row, the differences are preceding spaces (those before the leftmost asterisk) and number of asterisks.

Initialize the number of preceding space of the first row.
Initialize the number of asterisks for the first row.
As long as (number of asterisks is larger than zero)
begin
    Print spaces for number of preceding spaces times.
    Print asterisks for number of asterisks times.
    Print new line.
    Update number of preceding spaces for the next row.
    Update number of asterisks for the next row.
end

## Task F. Upside-down trapezoid or triangle

Write a program `trapezoid.cpp` that prints an upside-down trapezoid of given `width` and `height`.

However, if the input height is impossibly large for the given width, then the program should report, `Impossible shape!`

**Example 1:**

```
Input width: 12
Input height: 5

Shape:
************
 **********
  ********
   ******
    ****
```

**Example 2:**

```
Input width: 5
```

```
Input height: 3

Shape:
*****
 ***
  *
```

**Example 3:**

```
Input width: 12
Input height: 7

Impossible shape!
```

**Hint:**

You can start with the number of

```
spaces = 0;
stars = width;
```

On each line, print **that number of spaces** followed by **that number of stars**. After that, the number of spaces gets incremented by 1, while the number of stars gets decremented by 2:

```
spaces += 1;
stars -= 2;
```

To find out whether a shape is possible or not, do the following.

(1) The change of number of asterisks in adjacent lines is 2.
(2) If size is an odd number, the minimum number of asterisk is 1, otherwise, the minimum number of asterisks is 2.
(3) Find how many steps it takes to decrease from size to the minimum number of asterisks.
For example, suppose width is 5. To decrease from maximum number of asterisks 5 to the minimum number of asterisk 1, it takes (5-1)/2 = 2 steps. Hence, the maximum number of layers is 2 + 1 = 3. Suppose width is 5 and height is more than 3, the shape is impossible.

# Task G. Checkerboard (3x3)

Write a program `checkerboard3x3.cpp` that asks the user to input `width` and `height` and prints a checkerboard of 3-by-3 squares. (It should work even if the input dimensions are not a multiple of three.)

**Example 1:**

```
Input width: 16
Input height: 11

Shape:
***   ***   ***
***   ***   ***
***   ***   ***
   ***   ***   *
   ***   ***   *
   ***   ***   *
***   ***   ***
***   ***   ***
***   ***   ***
   ***   ***   *
   ***   ***   *
```

**Example 2:**

```
Input width: 27
Input height: 27

Shape:
***   ***   ***   ***   ***
***   ***   ***   ***   ***
***   ***   ***   ***   ***
   ***   ***   ***   ***
   ***   ***   ***   ***
   ***   ***   ***   ***
***   ***   ***   ***   ***
***   ***   ***   ***   ***
***   ***   ***   ***   ***
   ***   ***   ***   ***
   ***   ***   ***   ***
   ***   ***   ***   ***
***   ***   ***   ***   ***
***   ***   ***   ***   ***
***   ***   ***   ***   ***
```

```
   ***    ***    ***    ***
   ***    ***    ***    ***
   ***    ***    ***    ***
***    ***    ***    ***    ***
***    ***    ***    ***    ***
***    ***    ***    ***    ***
   ***    ***    ***    ***
   ***    ***    ***    ***
   ***    ***    ***    ***
***    ***    ***    ***    ***
***    ***    ***    ***    ***
***    ***    ***    ***    ***
```

Hint:
Label rows and columns by vertical blocks and horizontal blocks. Use 11 rows and 16 columns to illustrate the idea. Let h_block mean horizontal block and v_block mean vertical block.

| | h_block | 0 | | | 1 | | | 2 | | | 3 | | | 4 | | | 5 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| v_block | row\col | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 0 | 0 | h_block = 0<br>v_block = 0<br>print * | | | h_block = 1<br>v_block = 0<br>print space | | | h_block = 2<br>v_block = 0<br>print * | | | h_block = 3<br>v_block = 0<br>print space | | | h_block = 4<br>v_block = 0<br>print * | | | |
| | 1 | | | | | | | | | | | | | | | | |
| | 2 | | | | | | | | | | | | | | | | |
| 1 | 3 | h_block = 0<br>v_block = 1<br>print space | | | h_block = 1<br>v_block = 1<br>print * | | | h_block = 2<br>v_block = 1<br>print space | | | h_block = 3<br>v_block = 1<br>print * | | | h_block = 4<br>v_block =1<br>print space | | | |
| | 4 | | | | | | | | | | | | | | | | |
| | 5 | | | | | | | | | | | | | | | | |
| 2 | 6 | h_block = 0<br>v_block = 2<br>print * | | | h_block = 1<br>v_block = 2<br>print space | | | h_block = 2<br>v_block = 2<br>print * | | | h_block = 3<br>v_block = 2<br>print space | | | h_block = 4<br>v_block = 2<br>print * | | | |
| | 7 | | | | | | | | | | | | | | | | |
| | 8 | | | | | | | | | | | | | | | | |
| 3 | 9 | h_block = 0<br>v_block = 3<br>print space | | | h_block = 1<br>v_block = 3<br>print * | | | h_block = 2<br>v_block = 3<br>print space | | | h_block = 3<br>v_block = 3<br>print * | | | h_block = 4<br>v_block = 3<br>print space | | | |
| | 10 | | | | | | | | | | | | | | | | |

1. Omit the contents of last column due to space limit. But you can see yellow block has * while blue block has space.
2. Calculate v_block based on row.
3. Calculate h_block based on column.
4. When v_block and h_block have the same parity, print *, otherwise, print space. (How is this related with Task B. Checkerboard?)