This is a rather long lab. Don’t worry if you can’t finish all the problems. Pick 3 problems and focus on them.

1. Factorial Triangle

Write a ConsoleProgram named FactorialTriangle that asks the user to enter a number N, and then prints the value of n! for all n between 1 and N, as well as the definition of n! (that is, $1 \times 2 \times 3 \times \ldots \times n$) as shown below.

Enter a number: 8
1! = 1
2! = 1 \times 2 = 2
3! = 1 \times 2 \times 3 = 6
4! = 1 \times 2 \times 3 \times 4 = 24
5! = 1 \times 2 \times 3 \times 4 \times 5 = 120
6! = 1 \times 2 \times 3 \times 4 \times 5 \times 6 = 720
7! = 1 \times 2 \times 3 \times 4 \times 5 \times 6 \times 7 = 5040
8! = 1 \times 2 \times 3 \times 4 \times 5 \times 6 \times 7 \times 8 = 40320

2. Stars

Write a ConsoleProgram named Stars that prints out a sequence of ‘*’ characters in the form of a pattern as shown below. The program should ask the user for the size of the pattern to be printed (You can assume that the input size is always an odd number). The program should use nested loops and the output should look like the following.

Enter the size: 5
*****
* *
* *
*****

3. Diamond

Write a ConsoleProgram named Diamond that prints out a sequence of ‘*’ characters in a shape of a diamond as shown below. First, the program asks for the dimension of the diamond from the user. Assume that the input number is odd.

Enter a number: 9
*
***
*****
*******
*********
**********
***********
***********
**********
*****
*
4. **Headline (Question 4.16 in your textbook):**

Write a **GraphicsProgram** named **Headline** that makes a headline scroll from right to left in continuously. Recall that you can create a **GLabel** object as in

```
new GLabel("May the force be with you", xLocation, yLocation);
```

You can set the font of the text by sending it the `setFont` message: e.g. `label.setFont("Arial-12")`

Hint: You can obtain the width of a graphics object by sending it the `getWidth()` message. E.g: `label.getWidth()`.
5. **Pyramid (Question 4.11 in your textbook):**

Write a GraphicsProgram named *Pyramid* that produces an output similar to the following:

![Pyramid Diagram](image)

Try to center the pyramid in the window and use named constants for the following parameters:

- **BRICK_WIDTH** The width of each brick.
- **BRICK_HEIGHT** The height of each brick.
- **BRICKS_IN_BASE** The number of bricks in the base.

Use the `pause()` method to animate the order in which bricks are drawn.

When you're done, modify the program to get the output below:

![Pyramid Diagram](image)

Finally, modify your code to obtain

![Pyramid Diagram](image)

6. **Gravity (extra challenge)**

Add gravity to the *BouncingBall* program from the last lab.