6.23 *(Displaying a Diamond)* Write an app that displays the following diamond shape. You may use output statements that display a single asterisk (`*`), a single space or a single newline character. Maximize your use of iteration (with nested `for` statements) and minimize the number of output statements.

7.35 *(Recursive Sum Calculation)* Write recursive method `SumOfDigits(number)` that, when called, returns the *sum of digits* of the number passed.

For example, `SumOfDigits(234) = 2 + 3 + 4`. Assume that the number entered is greater than or equal to 10 but less than or equal to 99,999. The recursive step should use the relationship

```
((number%10) + SumOfDigits(number/10)).
```

The terminating condition occurs when the number is less than 10. Incorporate this method into an app that enables the user to enter the number.

8.19 *(Airline Reservations System)* A small airline has just purchased a computer for its new automated reservations system. You have been asked to develop the new system. You're to write an app to assign seats on each flight of the airline's only plane (capacity: 10 seats).

Display the following alternatives: `Please type 1 for First Class` and `Please type 2 for Economy`. If the user types `1`, your app should assign a seat in the first-class section (seats 1–5). If the user types `2`, your app should assign a seat in the economy section (seats 6–10).

Use a one-dimensional array of type `bool` to represent the seating chart of the plane. Initialize all the elements of the array to `false` to indicate that all the seats are empty. As each seat is assigned, set the corresponding element of the array to `true` to indicate that the seat is no longer available.

Your app should never assign a seat that has already been assigned. When the economy section is full, your app should ask the person if it's acceptable to be placed in the first-class section (and vice versa). If yes, make the appropriate seat assignment. If no, display the message `"Next flight leaves in 3 hours."`