CS 110 Lab Sheet 5

Lab cycle starting 30th October 2019, Normal Deadline, two weeks after your usual lab in that cycle.

Stage 1: While vs For

Convert the following while loop into a for loop. You may find it helpful to refer to the example programs in Chapter 4 ‘Loops' on Blackboard - particularly the DoubleInvestment and YearsInvestment ones.

```
int i = 0;
while (i < 100) {
     System.out.println(i);
     i++;
}
```

Stage 2: While vs For Part 2

In this example, we are using a for loop when we should be using a while loop - convert the loop to a while loop.

```
/* Count the number of times a user enters a number, until they
enter the number 50 */
Scanner in = new Scanner(System.in);
int count = 0;
for(int i = 0; count != 50; i++) {
     System.out.print(“Enter a number: “);
     count = in.nextInt();
     System.out.println(i);
}
System.out.println("Finished");
```

Challenge(ish)

In the program above (or your while version) change `int count = 0;` to `int count;` - why do you get a compiler error? *This is by far the easiest challenge task in all the lab sheets so you really should try to do it.*

Stage 3: Calculating Sum of Odd Numbers

Write a program that calculates the sum of odd numbers from 1 up to and including 19 using a loop - that is, 1 + 3 + 5 + … + 15 + 17 + 19. Remember what was said in the lectures about when you should use for and while loops and choose the right kind!

HINT: There are two ways of doing this. One uses an if statement inside a loop to check if a number is odd (remember that the remainder when you divide an odd number by 2 is 1, and for an even number 0 - the Java remainder operator is %). The other way does *not* need an if statement. For this, exercise either is fine - but if you think the if statement way is too obvious and easy for you, try to think what the other way would be (it's actually shorter).

Stage 4: Nested Loops

Write a program that accepts reads two integers from the user - one representing length and one representing height. It should then print out a square of *s length wide and height deep. E.g. if length is 8 and height is 5:

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

This will need two loops - one inside the other (*nested*).

Stage 5: Empty Boxes

Repeat the task 4 but print a box of *s instead. E.g

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

This - done in a straightforward way - needs four loops in total.

Challenge Task

Now do it again and generate triangles:

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

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

The tricky bit is that you won't be able to get all of the
lines apart from the top and bottom exactly the right length in all cases (when the height and width are not the same).

Assessed Task: Sum of Odd Numbers Again

Copy your sum of odd numbers program and extend it as follows. Instead of a fixed value of 19, the program should ask the user to enter a number oddSumMax and calculate the sum of odd numbers from 1 up to oddSumMax. However, the program should check that oddSumMax is actually odd. If it isn't the program should ask the user to enter another number instead. This will mean your final program should contain two loops.

Challenge Task

Write a program that inputs an integer and outputs it in binary. You will need a loop to do this. Remember that you can join Strings with '+' and that `num % 2` will either equal 1 or zero, where `num` is an integer.

Challenge Task

Write a program that inputs a string representing a binary number, and converts it into a decimal number. To do this you will need to find and read some of the documentation for Strings.