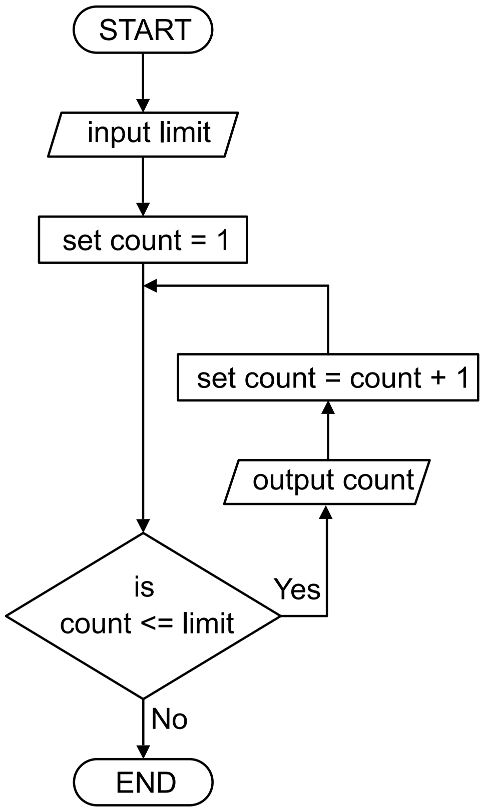
English name: ____________________

Iteration Assignment: `countTo`

Recall that when we studied flowcharts and pseudocode, we encountered iteration. We drew a flowchart that represented an algorithm that counted up to a value given by an input named `limit`. A solution for the flowchart is given to the right, and example pseudocode is given below.

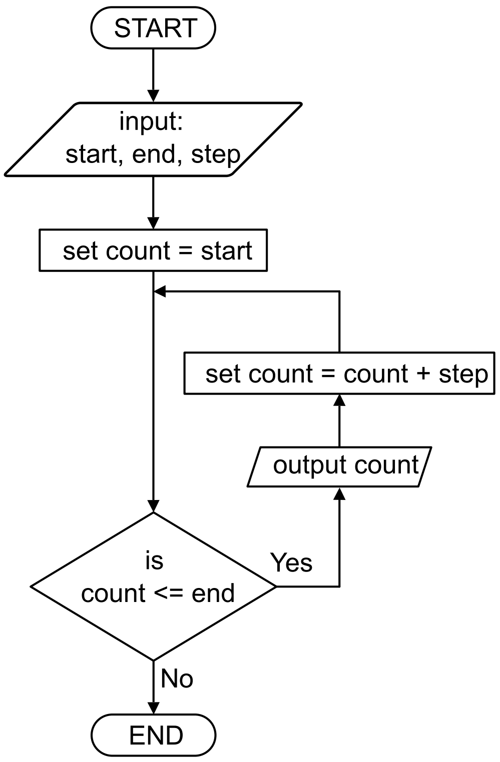
```
RECEIVE limit FROM KEYBOARD
SET count TO 1
WHILE count <= limit DO
    SEND count to DISPLAY
    SET count TO count + 1
END WHILE
```



The example pseudocode uses a `WHILE` loop; however, for definite iteration, it is common to use a `FOR` loop.

1. Write a Java method named **`countToWhile`** that has no return value, takes a parameter named `limit`, and uses a `while` loop to print to the screen the numbers from 1 to `limit`, inclusive, on a single line with numbers separated by a space character. Test your code by calling the method from a Java `main` method.
2. Write a method named **`countToFor`** to the same specifications as question (1), except that it is implemented using a `for` loop instead of a `while` loop. Test the method.

3. Write a method named **`countTo`** that takes three integer parameters: `start`, `end`, and `step`. This method is to use a `for` loop to count from the value given in `start` up to the value given in `end`, while counting by the value given in `step`.
   For example, if the parameters `start`, `end`, and `step` are set to`5`, `10`, and `2`, respectively, the method will produce the following output: "`5 7 9`".
   If the values given by the parameters are `6`, `27`, and `3`, the method will produce the output:
   "`6 9 12 15 18 21 24 27`".
   The algorithm is shown in the diagram to the right.
4. Write a method that ***overloads*** the **`countTo`** method, this method taking only two integer parameters: `start` and `end`. This method is to count from the value given in `start` up to the value given in `end`, while counting by a step value of `1`. This method must call the `countTo` method you wrote in part (3).
5. Write another method that ***overloads*** the **`countTo`** method, this method taking only a single integer parameter: `end`. This method is to count from `1` up to the value given in `end`, while counting by a step value of `1`. This method must call the countTo method you wrote in part (3).



Example test code is given on the next page.

Iteration Assignment: `countTo`

The following is an example of test code that you could use to test the methods you write for this assignment. The values printed do not necessarily need to be printed on a single line, as shown in the example output, as long as the output conforms to the specifications given in the assignment, above.

Example Test Code:

```
public static void main(String[] args) {
   int start = 5;
   int end = 16;
   int step = 3;
   System.out.print("countToWhile(10): ");
   countToWhile(10);
   System.out.print("countToFor(12): ");
   countToFor(12);
   System.out.print("countTo("+start+", "+end+", "+step+"): ");
   countTo(start, end, step);
   System.out.print("countTo("+start+", "+end+"): ");
   countTo(start, end);
   System.out.print("countTo("+end+"): ");
   countTo(end);
}
```

Output from the above test code that does conform to the specifications:

```
countToWhile(10): 1 2 3 4 5 6 7 8 9 10
countToFor(12): 1 2 3 4 5 6 7 8 9 10 11 12
countTo(5, 16, 3): 5 8 11 14
countTo(5, 16): 5 6 7 8 9 10 11 12 13 14 15 16
countTo(16): 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
```