foo (start an integer, end an integer which is larger than start)
    total = 0
    i = start
    While (i <= end)
        add i to total
        increment i
    Return total

1. What do you think this code does?
   It adds the integers from start to end, inclusive.
2. Write down a loop invariant you could use to show that this algorithm correctly does what you claim.
   After t times through the while loop, ...
   \[ i (or \ i_t) \ is \ start + t \]
   and total is \( (start) + \ldots + (i_t-1) \)
3. Suppose your loop invariant is true. How can you show the algorithm is correct based on this?
   The last time through the loop is when \( i = \text{end} \).
   Since the loop increments \( i \), when the program is over, \( i = \text{end} + 1 \). Subbing this in our invariant proves
4. Now, prove your loop invariant by induction. Let \( t \) be the number of times through the while loop.
   a. Base Case: Show the loop invariant is true before the while loop ever executes, when \( t=0 \).
      Before executing the loop, \( i = \text{start} \) and total = 0.
      This is correct because the sum of start up to start-1 is an empty sum with value 0.
   b. Inductive Hypothesis: Suppose the loop invariant is true after \( t \) times through the while loop.
   c. Induction Step: Now show that the loop invariant is true after \( t+1 \) times through the while loop.
      Suppose \( i_t = \text{start} + t \) and total \( t = (start) + \ldots + (i_t-1) \).
      Since the \( (t+1) \)-st time through the loop increments \( i \), \( i_{t+1} = i_t + 1 = \text{start} + (t+1) \) as desired.
      The loop also adds \( i_t \) to total, so
      \[
      \begin{align*}
      \text{total}_{t+1} &= \text{total}_t + i_t \\
                       &= (start) + \ldots + (i_t-1) + (i_t) \\
                       &= (start) + \ldots + (i_{t+1} - 1) \quad \text{as desired}
      \end{align*}
      \]
A robot stands at the origin of a coordinate plane. The robot has been equipped with four types of movement:

Northeast move in the direction (1,1)
Northwest move in the direction (-1,1)
Southeast move in the direction (1,-1)
Southwest move in the direction (-1,-1)

The robot starts at the origin and wants to reach the point (1,0) using any sequence of moves it knows how to do. Will the robot be able to reach its destination?

Invariant: After any number of moves, the sum of the robot's position is even.

Let $k$ = the # of moves
Base case: \( k=0 \) start at the origin so \( x+y=0 \) which is even.

Suppose after \( k \) moves, if \((x_k, y_k)\) is the robot's position, then \( x_k + y_k \) is even.

Show: After \( k+1 \) moves, if \((x_{k+1}, y_{k+1})\) is the robot's position, then \( x_{k+1} + y_{k+1} \) is even.

Case 1: The \( k+1 \)st move is a NE move:

\[
x_{k+1} + y_{k+1} = x_k + 1 + y_k + 1
\]
\[
= x_k + y_k + 2
\]

Case 2: NW move

\[
x_{k+1} + y_{k+1} = x_k - 1 + y_k + 1
\]
\[
= x_k + y_k
\]

Case 3: SE move

\[
x_{k+1} + y_{k+1} = x_k + 1 + y_k - 1
\]
\[
= x_k + y_k
\]
Case 4: SW move

\[ x_{k+1} + y_{k+1} = x_k - 1 + y_k - 1 \]
\[ = x_k + y_k - 2 \]
\[ \text{even} - 2 \]
\[ \text{even} \]

In all cases, the next move keeps the sum of the coordinates even, so the invariant is proved.

We can use the invariant to prove that the robot can't get to (1,0) since we've just seen that the \(x+y \) of the robot's position is always even but \(1+0 \) is odd.
Prove these two facts about convex polygons using induction.

1) The sum of the interior angles of any convex n-gon is \((n-2)\times 180\) degrees.

a. **Base Case:**

We know triangles have 180\(\)\(^{\circ}\) and
\[
N=3 \quad (3-2)\times 180 = 180, \text{ so the formula is correct in this case.}
\]

b. **Inductive Hypothesis:** Suppose ...

Suppose the sum of the angles of an \((n)\)-gon is \((n-2)\times 180\) degrees.

Consider any \((n+1)\)-gon, we can break it into an \(n\)-gon and a triangle like this: 

By the induction hypothesis, we know the \(n\)-gon has \((n-2)\times 180\)\(^{\circ}\) and we know triangles have 180\(\)\(^{\circ}\) and we know
\[
\frac{n(n-3)}{2} \quad \text{diagonals.}
\]

2) Any convex n-gon has \(n(n-3)/2\) diagonals.

a. **Base Case:**

\[
N=3 \quad \text{Triangles have no diagonals and } 3(3-3)/2 = 0 \text{ so the formula is correct in this case.}
\]

b. **Inductive Hypothesis:** Suppose ...

Suppose any convex n-gon has \(n(n-3)/2\) diagonals.

c. **Inductive Step:**

Consider any \((n+1)\)-gon. We can break it into an \(n\)-gon and a triangle as pictured:

The diagonals of the \((n+1)\)-gon are either diagonals of the \(n\)-gon or not. By the induction hypothesis, there are \(n(n-3)/2\) diagonals of the \(n\)-gon. There are \(n-2\) diagonals of the \((n+1)\)-gon that connect the vertex labeled \(n+1\) to vertices 2 through \(n-1\).
These are not diagonals of the n-gon. Further, the diagonal of the \((n+1)\)-gon that connects vertex 1 to vertex \(n\) is also not a diagonal of the n-gon. So, the number of diagonals of the \((n+1)\)-gon is

\[
\frac{n(n-3)}{2} + n - 2 + 1
\]

\[
= \frac{n^2 - 3n + 2n - 2}{2}
\]

\[
= \frac{n^2 - n - 2}{2}
\]

\[
= \frac{(n+1)(n-2)}{2}
\]

which is what we are trying to show.
Suppose you are designing a new Tetris piece. A Tetris piece is a region made up of square cells of the same size, aligned so that they could fit in a grid. Here are some example Tetris pieces:

![Tetris pieces diagram]

You would like to design your new Tetris piece to have perimeter 15. You can use any number of cells. How should you design your new piece?

It's impossible to have perimeter 15, because I will prove that all Tetris pieces have an even perimeter.

Induction on $n$, where $n = \# \text{ of cells}$

**Base case:** $n = 0$. A shape with no cells has perimeter 0, which is even.

(Could also do $n = 1$, where perimeter is 4.)

Assume that any Tetris piece with $n$ cells has an even perimeter. If we add one cell, in any position, we must show the perimeter of the new shape with $n+1$ cells is always even. There are several cases for where we could add a new cell, which we count by the number of edges it shares with
The existing region:

- 0 edges shared: increases perimeter by 4, keeping it even
  (note: this case is only needed if your base case was n=0)

- 1 edge shared: decreases perimeter by 1, increases it by 3 for net increase of 2, keeping it even

- 2 edges shared: keeps the perimeter unchanged, so still even

- 3 edges shared: decreases perimeter by 3, increase by 1 for net decrease of 2, keeping it even

- 4 edges shared: reduces perimeter by 4, so still even

Then, in all cases, the shape with n+1 cells has even perimeter, so we're done.