- We have considered correctness of assignment statements and if-statements

- Need to consider the correctness of loops

- The correctness of loops can be shown in two ways (we will look at each of these in turn).

- First, we need to understand a particular property of loops.

- Consider the example of adding a set of n numbers together

- $sum = 56 + 67 + 98$;

- $sum = a + b + c$;

- Deficiencies in these approaches?

- $sum = (a_1 + a_2 + \ldots + a_n)$;

- However the computer's arithmetic unit can only add 2 numbers at a time

- Proceed as follows:
  $sum = a_1 + a_2$; Step 1
  $sum = sum + a_3$; Step 2
  $sum = sum + a_4$;
  $\cdots$
  $sum = sum + a_n$;

- From step 2 on repeating the same process

- Needs to work for $n \geq 0$. Redo step 1.

- $sum = 0$; and

- $sum = sum + a_1$;

- Remember loops can iterate a varying number of times

- Something remains the same from iteration to iteration

  - This is called the invariant

- Consider some patterns of shapes

- What changes/ does not change?

- How did we identify the next shapes in the pattern?

- Loops: statements that repeatedly change the program variables

- To understand the loop and to determine the loop invariant, identify what is left *unchanged*

- Consider the example of adding n numbers

- Can you determine the invariant of the loop?

- An iterative implementation of the sum function

```
int totalSum(int arrayNums[])
{
    int i=0;
    int sum=0;
    while(i< arrayNums.length)
    {
        sum=sum+arrayNums[i]; i=i+1;
    }
    return sum;
}
```

- How can you prove that the loop is correct?
- 1st Approach:
- Use the while rule: $\frac{\{D \wedge I\} C \{I\}}{\{I\} \text{ while } D \text{ do } C \{\neg D \wedge I\}}$
- Show how the while rule holds for the loop on the previous slide (in lecture)
- We will see another approach to showing that a loop is correct but first we must look at induction and recursion

- What you should be able to do from this lecture:
  - Identify what remains unchanged when studying patterns
  - Identify the invariant for a loop
  - State the while rule for proving the correctness of a loop
  - Apply the while rule to a loop to show that it is correct