- In the last lecture we:
  - Constructed an iterative implementation of the function Sum to sum the numbers in a list.
  - Drew a flowchart for this iterative implementation.
  - Annotated the flowchart with assertions.
  - Proved by induction that the iterative implementation of the recursive definition was correct.

- In this lecture we look at a final example of:
  - Constructing a recursive definition of a function;
  - Implementing the recursive definition recursively;
  - Implementing the recursive definition iteratively;
  - Constructing a flowchart for the iterative implementation, annotating the flowchart with assertions and proving by induction that the iterative implementation is correct.

- The Fibonacci Numbers
- 0,1,1,2,3,5,8,13,21,34,55,89,144,...
- What is the pattern here?

- Recursive definition of function to return a specific number in the sequence:

  - Base Case: $Fib(0) = 0$, $Fib(1) = 1$

  - Recurrence Relation:
    $Fib(n) = Fib(n-1) + Fib(n-2)$

- Recursive Implementation:

```
int Fibonacci(int n)
{
    if (n==0)
       return 0;
    else if (n==1)
       return 1;
    else
       return (Fibonacci(n-1) +
                                 Fibonacci(n-2));
}
```

- An iterative implementation of Fib.

```
int Fibonacci(int n)
{
    int i=1;
    int fib1=0;
    int fib2=1;

    /* initialisation corresponding
                to the base case */

   int fib3;
   if (n==0)
     fib2=fib1;
   else {
        while(i<n)
        {
           fib3=fib1+fib2;
   fib1=fib2;
   fib2=fib3;
           // implementation of
           // recurrence relation
   i=i+1;
        }
   }
   return fib2;
}
```

- The following will be provided in the lectures:

  1. A flowchart for this iterative implementation.

  2. Annotate the flowchart with assertions.

  3. Prove the correctness of the iterative implementation using the assertions.