- Consider again the recursive definition of the factorial function (see previous lecture)

- Any inductively defined function has two parts:

  1. the specification of the values returned by the function for the basis

  2. the recurrence relationship between the values returned by the function for adjacent elements e.g. $n+1$ and n or n and $n-d$

- From the inductive definition we can derive:

  1. a recursive implementation where we go from what we want to what we know

  2. an iterative implementation where we go from what we know to what we want

- Recursion in programming implements a recursive definition

- A recursive(inductive) definition reflects the invariant of a loop more directly

- A recursive definition is often used to define a set with an infinite number of elements

- You may already have come across a recursive definition of an infinite set

- Consider a recursive implementation

- The two components of the recursive definition are combined into a single conditional statement based on:

  1. testing if the value of the argument is equal to a basis value

  2. applying the recurrence relationship and evaluating the inductively defined function again (for an argument closer to the basis value)

```
int fac(int n)
{
    if n==0 return 1; else return n*fac(n-1);
}
```

- Note the equivalence between the second part of the function and the recurrence relationship

- An iterative implementation of a function has three components:

  1. The initialisation which implements the base cases

  2. A loop which implements the recurrence relationship

  3. Declaration of extra local variables which in this example are:

     (a) The variable i which keeps track of our progress towards n

     (b) The variable result which holds the corresponding value of fac(i)

- An iterative implementation of the factorial function

```
unsigned int fac(unsigned int n)
{
    unsigned int i, result;
    i=0; result=1;
    /* initialisation corresponding
               to the base case */
    while(i<n)
    {
        i=i+1; result=i*result;
        // implementation of recurrence relation
    }
    return result;
}
```

- In previous lectures we have constructed iterative implementations (i.e loops) to solve some problems.

- Lets reconsider some of these problems and construct inductive definitions, iterative implementations, recursive implementations.

- Problems:

  1. Find the sum of the first n natural numbers

  2. Use repeated addition to evaluate multiplication

  3. Use multiplication to evaluate one number to the power of another