# Lecture #6: Recursion

**Reading**: Holmes, Chapter 12
**Handouts**: Factorial

1. Greetings and felicitations
2. What is recursion?
   - math: think induction
   - factorial: n! = n * (n-1)!, 0! = 1
3. Program stack
   - where arguments, locals, return addresses, state variables go
   - walk through 3! to show how to use it
4. Equivalent to iteration
   - show factorial; equivalence can be proven mathematically
5. Thinking about it
   - if you can change probem to a simpler case, you got it
   - need a terminal case
6. String length
   - break it down
   - show solution

# Factorial

This prompts you for an integer and prints the factorial. It does ***not*** check for overflow!

```
/*
 * program to compute a factorial
 * invocation:
 *    fact         run it
 * exit code: 0 always
 * author: Matt Bishop, bishop@cs.ucdavis.edu, 9/16/96
 */
#include <stdio.h>
#include <stdlib.h>

/*
 * compute a single-precision factorial
 */
int fact(int n)
{
      if (n < 0)   return(-1);    /* sanity check */
      if (n == 0)  return(1);     /* terminal case */
      return(n * fact(n-1));      /* recurse */
}

/*
 * the main program
 */
int main(void)
{
      int n;                      /* number to compute factorial of */
      register int nread;         /* number of numbers read */

      /*
       * read numbers and compute their factorial
       */
      while(printf("> "), (nread = scanf("%d", &n)) != EOF){
            switch(nread){
            case 0:        /* bad input */
                   printf("Need a non-negative integer\n");
                   break;
            case 1:        /* good input */
                   printf("\t%d\n", fact(n));
                   break;
            default:       /* can't happen (uh huh ...) */
                   printf("HUH? nread = %d?!?!\n", nread);
                   break;
            }
      }

      /*
       * skip to a new line and quit
       */
      putchar('\n');
      return(EXIT_SUCCESS);
}
```