**CS5900 Special Topics - Programming in C**

**Recursion in C**

Recursion is a programming technique that allows the programmer to express operations in terms of themselves. In C, this takes the form of a function that calls itself. A useful way to think of recursive functions is to imagine them as a process being performed where one of the instructions is to "repeat the process". This makes it sound very similar to a loop because it repeats the same code, and in some ways it *is* similar to looping. On the other hand, recursion makes it easier to express ideas in which the result of the recursive call is necessary to complete the task. Of course, it must be possible for the "process" to sometimes be completed without the recursive call. One simple example is the idea of building a wall that is ten feet high; if I want to build a ten foot high wall, then I will first build a 9 foot high wall, and then add an extra foot of bricks. Conceptually, this is like saying the "build wall" function takes a height and if that height is greater than one, first calls itself to build a lower wall, and then adds one a foot of bricks.

**Recursion** is the process of repeating items in a self-similar way. Same applies in programming languages as well where if a programming allows you to **call a function inside the same function that is called recursive call of the function** as follows.

void recursion()

{

recursion(); /\* function calls itself \*/

}

int main()

{

recursion();

}

The C programming language supports recursion, i.e., a function to call itself. But while using recursion, programmers need to be careful to define an exit condition from the function, otherwise it will go in infinite loop.

Recursive functions are very useful to solve many mathematical problems like to calculate factorial of a number, generating Fibonacci series, etc.

## Number Factorial

Following is an example, which calculates factorial for a given number using a recursive function:

#include <stdio.h>

int factorial(unsigned int i)

{

if(i <= 1)

{

return 1;

}

return i \* factorial(i - 1);

}

int main()

{

int i = 15;

printf("Factorial of %d is %d\n", i, factorial(i));

return 0;

}

When the above code is compiled and executed, it produces the following result:

Factorial of 15 is 2004310016

## Fibonacci Series

Following is another example, which generates Fibonacci series for a given number using a recursive function:

#include <stdio.h>

int fibonaci(int i)

{

if(i == 0)

{

return 0;

}

if(i == 1)

{

return 1;

}

return fibonaci(i-1) + fibonaci(i-2);

}

int main()

{

int i;

for (i = 0; i < 10; i++)

{

printf("%d\t%n", fibonaci(i));

}

return 0;

}