ECE 220 REVIEW

And we’re back......
To understand recursion, one must first understand recursion.

-Stephen Hawking
My personal Favorite:
To loop is human, to recurse is divine
In programming terms, it basically means there is a function that just calls itself. That is it, no magic to it. The hard part is how to call that function.
In layman terms:
Read this sentence and do what it says twice.
Fundraising: Iteration vs Recursion

Problem: You have to collect $1000 for charity. You can only ask for a penny from one person.

Iteration: Either you go to 100,000 people and ask for a penny each. (Ouch!)

Recursion: If you are asked to collect a penny, give a penny to the person who asked you for it. Otherwise:

1) Visit 10 people and ask them to each raise 1/10th of the amount of money that you have been asked to raise.
2) Collect the money that they give you and combine it into one bag
3) Give it to the person who asked you to collect the money.
Basic Example:
The Fibonacci Series

\[ \text{Fib}(N) = \text{Fib}(N-1) + \text{Fib}(N-2) \]
\[ \text{Fib}(0) = 1 \]
\[ \text{Fib}(1) = 1 \]
IMPORTANT:

Always REMEMBER TO HAVE A BASE CASE!!

Only 3 things matter:

1) BASE CASE
2) WHAT YOU Process
3) THE RECURSIVE Call
Let’s Do SOME Questions
#define LIMIT 1000
void fun2(int n)
{
    if (n <= 0)
        return;
    if (n > LIMIT)
        return;
    printf("%d ", n);
    fun2(2*n);
    printf("%d ", n);
}
If we call fun2(100) ; it prints 100,200,400,800,800,400,200,100
```c
int fun(int a[], int n) {
    int x;
    if (n == 1)
        return a[0];
    else
        x = fun(a, n-1);
    if (x > a[n-1])
        return x;
    else
        return a[n-1];
}

int main() {
    int arr[] = {12, 10, 30, 50, 100};
    printf("%d ", fun(arr, 5));
    getchar();
    return 0;
}
```
We get 100 as it finds the maximum number of the array.
EASY Question-3

A) Compute SUM of natural numbers upto N.
B) Fibonacci
C) TRibonacci
Problem 3: Recursion on MT2_Practice_Midterm_1
MeDium/HARD Question-2

Write Merge-Sort
MergeSort(arr[], l, r)
If r > 1

1. Find the middle point to divide the array into two halves:
   middle m = (l+r)/2
2. Call mergeSort for first half:
   Call mergeSort(arr, l, m)
3. Call mergeSort for second half:
   Call mergeSort(arr, m+1, r)
4. Merge the two halves sorted in step 2 and 3:
   Call merge(arr, l, m, r)