Discussion Section Notes for the Week of 9/24/01

Why do we use functions?

A. Modularity Example — This function was designed to sum a sequence of numbers from one to the input value. Since it is very general, it can be reused.

```c
int SumNumbers (int num)
{
    int answer;
    int i;

    answer = 0;
    for ( i = 0; i <= num; i++ )
    {
        answer += i;
    }

    return (answer);
}
```

B. Hiding Details — This function is not general enough to be reused, but it does hide details and make code more readable.

```c
void PrintAnswer (float result)
{
    printf ("The result was %f.\n", result);
    return;
}
```
Top–Down Design

A. Consider the problem of making a peanut butter and jelly sandwich:

```c
int main()
{
    GetIngredients();
    GetUtensils();
    MakeSandwich();
    PutAwayIngredients();
    WashUtensils();
    PutAwayUtensils();
    EatSandwich();

    return 0;
}

void GetIngredients()
{
    GetBread();
    GetPeanutButter();
    GetJelly();
}
```

Notice how the task is broken up into smaller and smaller pieces.

B. A Program Stub contains rough versions of all functions, including their parameter lists, although it does not contain working implementations.

```c
int AverageOfNumbers (double a, double b)
{
    printf("Function AverageOfNumbers just found the average\n");
    printf("of %f and %f\n", a, b);
    return 5;  /* For testing, we can assume the average is 5 */
}
```

C. Drivers are used to test implementations of the functions without testing the whole program.

```c
int main()
{
    /* We haven’t written our real main function yet, but we need to test
    the function AverageOfNumbers */
    int myAverage;
    myAverage = AverageOfNumbers (5, 7);
    printf("AverageOfNumbers returned value %f\n", myAverage);
    /* The value 6 should be printed */
    return 0;
}