Quiz 3 Sample Solutions

1) Write a function that calculates and returns the n^{th} Triangle Number, where n is the input parameter. You may assume that n is guaranteed to be positive. Note that n^{th} Triangle Number is the sum of the first n positive integers. For example, the 5^{th} Triangle Number is 15, since 1 + 2 + 3 + 4 + 5 = 15. (Note: You will get **no** credit on this question if you put a printf in this function.) Please fill out the function definition provided below:

```
int triangle(int n) {

    int i, sum = 0;

    for (i=1; i<=n; i++)
        sum += i;

    return sum;
}
```

2) The file "numbers.txt" has exactly 10000 non-negative integers in it, one per line. Complete the program below so that it opens the file, reads in all the integers, determines how many of those integers end in each digit, prints this information out, and closes the file. (See the document camera projection for a sample for clarification.)

```
#include <stdio.h>

int main() {

    FILE* ifp = fopen("numbers.txt", "r");
    int i, units[10], num;

    for (i=0; i<10; i++)
        units[i] = 0;

    for (i=0; i<10000; i++) {
        fscanf(ifp, "%d", &num);
        units[num%10]++;
    }

    for (i=0; i<10; i++)
        printf("%d %d\n", i, units[i]);
    fclose(ifp);

    return 0;
}
```

3) Write a function that takes in pointers to two integers and modifies the integers by storing the sum of the original integers in the first integer variable and the difference of the two integers in the second variable. For example, if ptrX points to a variable storing 7 and ptrY points to a variable storing 3 before the function is called, after the function completes, ptrX should point to the same location but that location should store 10 and ptrY should point to the same location but this location should store 4.

```
void sumdiffchange(int *ptrX, int *ptrY) {
    int sum = *ptrX + *ptrY;
    int diff = *ptrX - *ptrY;
    *ptrX = sum;
    *ptrY = diff;
}
```

4) Write a function that takes in an integer, representing a social security number and adds a random number in between 1 and 10, inclusive, to it and returns that value as the next social security number to assign. Assume the random number generator has been seeded and all necessary includes have been made.

```
int nextSSN(int curSSN) {
    return curSSN + 1 + rand()%10;
}
```

5) Write a function that takes in the current social security number and prints out the next n social security numbers assigned, by calling the nextSSN function n times. Print out one number per line on n lines and nothing else.

```
void printSSNs(int curSSN, int n) {

    for (i=0; i<n; i++) {
        curSSN = nextSSN(curSSN);
        printf("%d\n", curSSN);
    }
}
```

6) Complete the function below so that takes in an integer array and its length, returns 1 if the array stores **only** negative numbers, and returns 0 if the array stores at least one number that is not negative. (Note: 0 is NOT a negative number.)

```
int allNeg(int array[], int length) {

    for (int i=0; i<length; i++)
        if (array[i] >= 0)
            return 0;
    return 1;
}
```

7) What is the output of the following program?

```
#include <stdio.h>
int f(int a, int* b);

int main() {
    int a = 3, b = 7, c = 4;
    c = f(b, &a) + 3;
    printf("a=%d, b=%d, c=%d\n", a, b, c);
    b = f(a, &c);
    printf("a=%d, b=%d, c=%d\n", a, b, c);
    return 0;
}

int f(int a, int* b) {
    *b = (a+3)*2 + (*b)%4;
    a = *b - a%6;
    printf("a=%d, b=%d\n", a, *b);
    return 2*a - (*b);
}
```

a=22, b =23
a=23, b=7, c=24
a=47, b=52
a=23, b=42, c=52

8) An inversion in an array, **arr**, is when a pair of values in the array such that arr[i] > arr[j] but i < j. (Essentially it's a pair of values that are out of order.) For example, in the array [6, 3, 8, 1], the inversions are (6, 3), (6, 1), (3, 1) and (8, 1). For each pair, the first number appears somewhere to the left of the second number (in a lower index), but is bigger than the second number. Complete the function below so that it takes in an integer array, **arr**, and its length, **length**, and returns the number of inversions in the array. You may assume that the array size is less than 10000, so that the number of inversions fits in an integer.

```
int numInversions(int arr[], int length) {

    int res = 0;
    for (int i=0; i<length; i++)
        for (int j=i+1; j<length; j++)
            if (arr[i] > arr[j])
                res++;

    return res;
}
```