1. Implement Programming Exercise 5-Chapter 16 (Gaddis), pp.1022. Use integers and doubles in your test program. Output should be user friendly.

Name the program: TestTotalTemplateXX.cpp, where XX are your initials.

2. Write a recursive function that finds the median of an array of numbers. Then, write a C++ program to test your recursive function. Make sure the function can handle even and odd number of elements. The function must be recursive and not use a loop. Use the following two arrays in the test program to test your recursive function.

```
int[] odd = {1, 2, 3, 4, 5, 6, 7};
int[] even = {1, 2, 3, 4, 5, 6, 7, 8};
```

No sort is needed in your program since the numbers above are already in sorted order. Output should look similar to below.

```
Given the array: 1 2 3 4 5 6 7
The median is: 4

Given the array: 1 2 3 4 5 6 7 8
The median is: 4.5
```

Name the program: RecMedianXX.cpp, where XX are your initials.

Hint: Here is a non-recursive C function that finds the median of an array of numbers using pointer notation.

```
double median(int *arr, int num)
{
    double med;

    if (num % 2 == 0)    // Is number of elements even?
    {
       int mid1 = num / 2, mid2 = (num / 2) - 1;
       med = ((*(arr + mid1) + *(arr + mid2)) / 2.0);
    }
    else
       med = *(arr + (num / 2));
    return med;
```

}

3. Implement Programming Exercise 13-Chapter 15 (Gaddis), pp.988. Use the classes as mentioned in the text. Place all code into one file. Output should be user friendly.

Name the program: TestABCShapeXX.cpp, where XX are your initials.

4 (**). Write a C++ program to count pixels on a grid. The data are in a two-dimensional grid of cells, each of which may be empty (0) or filled (1). The filled cells are connected to form a blob (an object). A blob can consist of one or more ones connected horizontally, vertically, or diagonally. Thus a file consisting of all ones represents a single (very large) blob. The file begins with a line containing a single non-negative integer N that indicates the (square) dimensions of the grid (N x N). The value of N will never exceed 50. The next N lines will contain N characters. Output will consist of the file read in, the size of each blob in ascending order and the total number of blobs in the picture. If a grid contains no blobs, output the phrase: No blobs exist. Input the file name from the keyboard and check for valid files. Output should look similar to below.

```
15
000001111010000
000000100010000
100000010010001
000000011100000
000000000000000
000011000000000
000000000000000
001111000001000
000110000001000
000000100001000
000001100001100
000000100001111
000001111111111
000010000001111
000100000001111
```

```
Enter the file name: blob.txt

00000111101 0000
0000001 000 10000
1 0000001 00 1000 1
0000000 111 00000
000000000000000
0000 11 000000000
000000000000000
00 1111 000001000
000 11 0000001000
0000001 00001000
00000 11 0000 11 00
0000001 0000 1111
00000 1111111111
0000 1 000000 1111
000 1 0000000 1111
```

The sample above with the six areas highlighted. The input file will have spaces between the numbers.

Name the program: BlobCounterXX.cpp, where XX are your initials.