1. (15 points) Write a recursive function that adds the first $n$ values in an array containing $n$ entries.

Solution:

```c
int RecAdd( int a[], int i, int n )
{
    if ( i<n )
        return a[i] + RecAdd( a, i+1, n );
    else
        return 0;
}
```

2. (15 points) Write a recursive function that computes and returns the value of $m + n$ for non-negative integers $m$ and $n$. The only arithmetic operations you can use are $++$ and $--$. You can also compare values to 0.

Solution:

```c
int add( int m, int n )
{
    if ( n == 0 )
        return m;
    else
    {
        m++;
        n--;
        return add( m, n );
    }
}
```

Here's an extra challenge problem. Do not turn in solutions.

Write a non-recursive version of the function $c_{n,k}$ from Friday’s notes. (Hint: make a two-dimensional array and record results in it.)
**Solution:** The idea is to dynamically allocate a two-dimensional array of size \( n \times k \), initialize row 0, and then gradually fill in values.

```c
int c_n_k( int n, int k )
{
    // allocate the matrix (it is triangular)
    int i, j;
    int ** matrix;
    matrix = new int* [ n+1 ];
    for ( i=0; i<=n; ++i ) matrix[i] = new int[i+1];

    // fill base case of 0
    for ( i=0; i<=n; ++i ) matrix[i][0] = 1;

    // fill in the matrix
    for ( int i=1; i<=n; ++i )
    {
        int ubound = i<=k ? i-1 : k;
        for ( j=1; j<=ubound; ++j )
            matrix[i][j] = matrix[i-1][j] + matrix[i-1][j-1];

        // The other base case
        if ( ubound < k )
            matrix[i][i] = 1;
    }

    // save the result
    int result = matrix[n][k];

    // delete the matrix
    for ( i=0; i<=n; ++i ) delete [] matrix[i];
    delete [] matrix;

    return result;
}
```

There are other ways to compute this value, but they are beyond our scope.