LAB #6 – Iteration vs. Recursion

Practice Recursion:
You will practice writing and timing iterative vs. recursive functions. You will use the following code to time your iterative versus recursive solution to the following problem.

```cpp
#include <sys/time.h>
#include <cstdlib>
using std::cout;
using std::endl;

int main() {
    typedef struct timeval time;
    time stop, start;

    gettimeofday(&start, NULL);

    //Time your iterative or recursive function here.

    gettimeofday(&stop, NULL);

    if(stop.tv_sec > start.tv_sec)
        cout << "Seconds: " << stop.tv_sec-start.tv_sec << endl;
    else
        cout << "Micro: " << stop.tv_usec-start.tv_usec << endl;
    return 0;
}
```

In statistics, the formula for computing the number of ways of choosing \( r \) different things from a set of \( n \) things is the following:

\[
C(n, r) = \frac{n!}{(r! \times (n-r)!)}
\]

We already defined the factorial function as:

\[
n! = n \times (n-1) \times (n-2) \times \ldots \times 1
\]

Write an iterative and recursive version of the formula for \( C(n, r) \). Time each of these solutions for \( r=3, n=20 \) and \( r=10, n=1000 \).
What do you notice about the times for these two different solutions? What do you notice about the output/calculation for the two different inputs of \( r \) and \( n \)? What happens in the different solutions if \( n \) is 40000? Why?

**Discuss your answers to these questions with the lab TA for full lab credit.**

**Extended Learning:**
You can practice taking in **command line arguments**. This means when you run your program, you supply your commands at the time of execution, as in Unix commands, i.e. `ls –al`. In this lab, you can practice with taking in command line arguments for the \( n \) and \( r \) values in your previous lab. **You can decide to supply them in a specific order, or more realistically, supply them in any order. Use the options --r and --n to denote when the \( n \) and \( r \) are being entered, i.e. `./combinations --n 100 --r 3`.**

In order to read command line arguments, you need to alter your int main() function. You need to add `int main(int argc, char *argv[])`. The \( argc \) parameter is the count of arguments passed to the program, including the program name, i.e. \( argc \) is 5, and you can access the name of the program and arguments provided to the program using \( argv \), which contains the values of the arguments including the name of the program, i.e. \( argv[0] \) points to `./combinations`. The \( argv \) parameter is a 2-d array, i.e. \( argv[1][0] \) and \( argv[1][1] \) refer to the -- and n characters.

Now, read the \( n \) and \( r \) values using command line arguments!