Homework Assignment 3

1) Submit as `Assignment_03_1.cpp` using the provided template according to the instructions. Further details about formatting the output of your program are provided in the template file.

A palindrome is a sequence of symbols that is the same when reversed. For example the sentence, "A man a plan a canal Panama", has the same character sequence in the reverse order (ignoring spaces and case). Write a program to reverse a string constant and test if its sequence is equal to the forward direction, as required by a
palindrome.

2) Submit as `Assignment_03_2.cpp` using the provided template according to the instructions. Further details about formatting the output of your program are provided in the template file.

Later in the course we will have to find an efficient algorithm to solve the following problem: given a set of N 3D points $p_{1,}p_2, \dots, p_N$, compute the distance from an arbitrary 3D point p, to the closest point in the set. It is not necessary to identify which point from the set is the closest to the given point. We only want to know the distance. If this computation has to be performed for a single point p, or even for a small number of different points, the most efficient way of measuring the closest distance is to measure the distance $\|p - p_k\|$ to each of the N points of the set, and to determine the minimum. Note that the complexity of this computation is linearly proportional to the number of points N in the set, and it can be very time consuming when the set of points is large. If the computation has to be performed many times, say more than N, more efficient data structures and algorithms must be used. We are not going to solve this problem yet. Here we explore a simple special case.
Create a 10x10x10 three-dimensional array on the heap of type double. Set the diagonal elements to 0 (i.e., $N = 10$, and $p_k = (k, k, k)$ for $k = 1,2, \dots ,10$). Fill the other elements of the three-dimensional array with the distance to the nearest diagonal element, which in this special case can be computed analytically. Test your implementation by determining the distance from the eight corners of the array to the nearest diagonal element. Access these locations using only pointer arithmetic and the dereference operator, *. Delete the array from the heap after testing.

3) Submit as `Assignment_03_3.cpp` using the provided template according to the instructions. Further details about formatting the output of your program are provided in the template file.

Pascal's triangle produces the set of binomial coefficients as shown below,

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1

The values of the n^{th} row ($n \geq 0$) correspond to the coefficients of the polynomial $(x + y)^n$ when it is expanded as a linear combination of monomials. For example, if the polynomial $(x + y)^4$ is expanded into monomials the result is

$$(x+y)^4 = (1)x^4 + (4)x^3y^1 + (6)x^2y^2 + (4)x^1y^3 + (1)y^4$$

i.e., the coefficients are the last row of the triangle above. This example shows why these coefficients are called binomial coefficients. In general, the following notation is used for binomial coefficients

$$\binom{n}{k} = \frac{n!}{k!\,(n-k)!}$$

for $0 \leq k \leq n$, so that

$$(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^k y^{n-k}$$

It is clear from this definition that

$$\binom{n}{n-k} = \binom{n}{k}$$

for $0 \leq k \leq n$. These numbers can also be defined recursively by the following rules

1. $\binom{n}{0} = \binom{n}{n} = 1$ for all $n \geq 0$.
2. $\binom{n+1}{k+1} = \binom{n}{k} + \binom{n}{k+1}$ for all $n \geq 0$ and $0 \leq$ k < n

Design a way to store Pascal's triangle on the stack (local scope declaration) using a linear array. Note that the first row requires one memory location, the second row requires two, the third row requires three, and so on. Your memory representation should be able to store any lower triangular matrix without wasting storage. Given a pair (n, k) , where $n \geq k \geq 0$, you need to derive a formula to compute the location in the linear array where the corresponding binomial coefficient is stored. A square array is not the solution to this problem. Print the first 16 rows to cout and compare with the example at

http://en.wikipedia.org/wiki/Pascal's_triangle .

4) Submit as `Assignment_03_4.cpp` using the provided template according to the instructions. Further details about formatting the output of your program are provided in the template file.

Determine if your computer CPU is big-endian or little-endian by implementing a computation that will reveal the byte order of the representation of numerical data types. Verify your conclusion by using your debugger to examine the contents of memory locations holding a known unsigned short integer.

5) Submit as `Assignment_03_5.cpp` using the provided template according to the instructions. Further details about formatting the output of your program are provided in the template file.

Following are the first few prime numbers: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199. Store these numbers in a one- dimensional array. Write a program to find the

nearest prime number to a given query value by carrying out binary search.

Binary search operates on a sorted list of numbers. A query number is compared with the middle number of the range of stored values. If the query is less than the middle number, the new upper bound on the range is the next stored number below the middle. If the query is greater than the middle number then the lower bound on the range is the next stored number above the middle. If the query equals the middle number then the answer is found. A more complete description of binary search is given at

http://en.wikipedia.org/wiki/Binary_search

Consider in your program issues such as the case where the query is outside the range of the stored numbers. Test your program using all the even numbers from 2 to, and including, 200.