CMPUT 350 Lab 7 Prep Problems

---

1. Write function template `selection_sort` that takes an iterator range and a comparison functor as parameters and sorts the range according to the functor using selection sort (i.e., repeatedly choosing the minimum of the remaining values and swapping it with the current element), so that the following code works:

```
...
struct MyLess { ... };
struct MyGreater { ... };
vector<int> v{3,5,4};       // vector contains 3 5 4
selection_sort(begin(v), end(v), MyLess());        // 3 4 5
selection_sort(begin(v), end(v), MyGreater());     // 5 4 3
```

---

2. Write a program that reads integers from `stdin` and prints all encountered values exactly once in non-decreasing order to `stdout`.

```
E.g.:  5 3 4 1 4 3 => 1 3 4 5
```

Can you give a non-trivial upper bound on the runtime of your algorithm depending on the number of ints in the input (n) ?

Hint: in what order are elements visited when traversing a `std::set<int>` ?

3.

a) Write function `long long choose(int n, int k)` that for $k \geq 0$ and $k \leq n$ computes

- $\text{choose}(n, k) = 1$, for $k = 0$ or$k = n$
- $\text{choose}(n, k) = \text{choose}(n-1, k-1) + \text{choose}(n-1, k)$, otherwise

[also known as binomial coefficient -

$$\binom{n}{k}$$

or the number of ways of choosing k objects out of n; above recursion is known as "*Pascal's Triangle*"]

b) Write function `choose_m` that speeds up computing `choose(n,k)` by memoization (i.e., storing and recalling already computed function values in a `std::map`, see lecture cpp notes for an example)

c) Lastly, determine the highest value of `n` for which `choose_m(2*n, n)` computes the correct value. For this, consider using Boost's arbitrary precision integer type `cpp_int` like so:

```
#include <boost/multiprecision/cpp_int.hpp>
using namespace boost::multiprecision;
...
// look - HUGE numbers!
cpp_int i = 2;
cpp_int j("1000000000000000000000000000");
cout << i*j << endl;   // prints 2000000000000000000000000000
```