CMPUT 350 Lab 6 Exercise Problems

Rules:

- You can use all course material and man pages, but no other information such as web pages, books, or written notes. Using other information sources during the exercise constitues cheating.
- Your programs must compile without warning using

  ```
  g++ -Wall -Wextra -Wconversion -Wsign-conversion -O -g -std=c++17 ...
  ```

  In case there are compiler warnings or errors, marks will be deducted.
- Test your programs with different values. For now, the speed of your program is irrelevant. So don't spend time on optimization
- You must check for the appropriate preconditions/postconditions. Your program shouldn't crash or have undefined behaviour (**hint**: use asserts)!
- Your programs must be well structured and documented. Use ctrl-x t in Emacs to pretty-print it. Marks are assigned to functionality, program appearance, and comments.
- In case your program hangs, use ctrl-c to terminate it.
- Remember that you need to include the appropriate header files. To find out which ones you need for specific functions such as printf, use then man command.

Submit your solution files `Stack.h` `mainStack.cpp` `mainSum.cpp` on eClass under "Lab 6 / Submission".

Important: Submit often (the system will only accept solutions submitted before 16:50)

1. [43 marks]

   a) In file `Stack.h` write a template class `Stack<T>` that stores elements of type `T` and supports the following public class methods:

   ```
   empty  // returns true iff stack contains no element
   top    // returns reference to top element (pre-cond: !empty)
   push   // push an element on top of the stack
   pop    // remove the top element from stack (pre-cond: !empty)
   clear  // remove all elements
   ```

   In your implementation use private inheritance from `std::vector<T>`. All Stack member functions except `clear()` must run in amortized constant time

   b) In `Stack.h` implement

   ```
   template <typename T>
   std::ostream &operator<<(std::ostream &os, const Stack<T> &s);
   ```

   that prints a stack to an output stream (in one line, starting with the top element, using a single space (' ') as separator). You may assume that stack elements can be printed using `cout << ...`

Note that above `operator<<` can only use `Stack`'s public interface (empty/top/push/pop/clear)

c) In `Stack.h` also write global template function "`reverse_stack`" that reverses the order of elements in a given Stack. Think about how to do this using only the public interface.

d) In file mainStack.cpp write test code that checks all functions you wrote. The weight of correct implementation and testing is equal in this exercise.

IMPORTANT! Even if you haven't implemented some functions, you can still write test code for them!

Also check pre-conditions with assert, use const as often as possible, and ensure that you can push expression results (e.g., `Stack<int> s; s.push(3+5);`)

Finally, make sure that your code doesn't leak memory!

---

2. [10 marks] In file `mainSum.cpp` write and test template class `Sum` that for $n \geq 0$ computes `1+2+..+(n-1)+n` at **COMPILE TIME**

```
int main() {
    cout << Sum<0>::value  << endl;          // 0
    cout << Sum<10>::value << endl;          // 55
    cout << Sum<20>::value << endl;          // 210
    cout << Sum<100>::value << endl;         // 5050
    // etc.
    return 0;
}
```