Lab 5

Week of 2/1

Partnering Up

- One partner does *mystring*
- One partner does modified *poly* class from last week
  - instead of fixed size array, we are using dynamic array, as such the private members have changed
  - now that we are using dynamic array, copy constructor and assignment operator need to be written
  - instead of using assert to make sure that the size of array doesn't exceed capacity, use reserve function to add to the size of the array
  - Write destructor since dealing with dynamic memory
- Partner who wrote *poly* last week must write *mystring* this week

String

- Implement a string class
- Dynamically allocate memory for the string
- Private variables
  - char *characters; // Pointer to first char in string
  - size_t allocated; // Number of chars allocated for string
  - size_t current_length; // Number of chars currently in string at any given time
- 3 constructors
- Destructor
  - Need to free memory since the string is dynamically allocated
  - Use *delete*

Provided Files

- mystring
  - mystring.h
    - Implemented for you all this week
    - length() is also already written
  - str_demo.cpp
    - Do not need to edit this file
    - Will be used for demo
    - **Very very** limited testing of your string class
- poly → Make sure to download most recent version
  - poly.h (modified from last week)
  - Intr_poly_tester.cpp (do not edit)
  - polygif.cpp (do not edit)

Tips

- Reserve() should be the first modification function that you write *(both)*
  - Can use this in the constructors along with the insert functions and writing reserve() first can make those functions easier to write and understand
- Use <cstring> functions *(mystring)*
  - #include <cstring>
- Test often when writing functions *(both)*
  - Write insert() and then test it with a few different cases to make sure that it works correctly
- << operator *(both)*
  - No need to cout << endl
- >> operator *(mystring)*
  - *ins* is a string that you will be traversing
  - Think of *ins* as the input and you are tasked with taking *ins* and adding it to *target*

Submission Files

- Turn in everything that you did personally
  - Report that you wrote about your partner's code
  - Test cases that you wrote for your partner
  - Your own implementation code
- No need to turn in your partner's work

Don’t forget

- Demo code to me
  - Either today or next week
  - **Must compile and run on linux servers**
- Submit code to camino by the end of next lab
- Comment code
  - Loops and conditionals
- File with description of lab is on Camino
- Check google sheet to make sure that I didn’t forget to check you off for a demo