Jan 25th lecture notes

* Homework 1
  + Double d[100];
  + Di <-> d[I];
  + Double average (double \*d, int size) (passes the pointer of the array to the function)
  + Main()
    - Double d[100];
    - Double a = average(d,n);
* Lecture and Lab 3
  + Lab 3 String
    - To mirror the STL string we have to change the class to capital S
    - String.h
    - Class String
    - {
      * Private:
        + Char \*buffer;
        + Int size; (running size of the array, will not null terminate)
      * Public:
        + String(); (in main its called when String s; is typed)

//default constructor

* + - * + String(const char\*) (main ex: String t("abc");
        + String(const String &); (main ex:String w(t); calling prev)

// copy constructor

* + - * + ~String();
        + Int length();
        + Char & operator[](const int); -> cout << t[5]; or t[5]='b';
        + String operator=(const String &); t = s = w;

//sets s to w then the new s is assigned to t;

* + - }
  + Default constructor – String.cpp
    - #include "String.h"
    - #include <cstring>
    - String::String()
    - {
      * Buffer=null; or 0 same thing
      * Size = 0;
    - }
  + First overload
    - String::String(const char \*p)
    - {
      * Size = strlen(p);
      * Buffer = new char[size];
      * For (int I = 0; I<size; I++)
        + Buffer[I] = p[I];
    - }
  + Second overload
    - String::String(const String &source)
    - {
      * // W is implicit and T is explicit. Implicit because it’s the object calling the method. Expliciti because its coming in as a parameter
      * Size = source.size();
      * Buffer = new char[size;
      * For (int I= 0; I < size; I++)
        + Buffer[I] = source.buffer[I];
      * // try to write the for loops with just pointers.
    - }
  + Destructor
    - String::~String()
    - {
      * Delete[] buffer;
    - }
  + Operator overload []
    - Char & String:: operator[](const int I;)
    - {
      * Assert(I<size and I>=0);
      * Return buffer[I];
    - }
  + Operator overload =
    - Char & String::operator=(const String &source)
    - {
      * Delete[] buffer;
      * Size = source.size();
      * Buffer = new char [size];
      * For (int I= 0; I < size; I++)
        + Buffer[I] = source.buffer[I];
    - }
  + Operator overload <
    - bool String::operator<(const String &)
    - Friend bool operator (String, String); Similar to string compare.
* Stack of Activation records
  + Int f(int x)
  + {
    - Int y;
    - Y=x;
    - Y++;
    - Return y;
  + }
  + Void g()
  + {
    - Int n
    - Cin >> n;
    - Int m = f(n);
    - Cou << m;
  + }
  + Main()
  + {
    - Int var;
    - Cin >> var;
    - Cout << f(var);
    - G();
  + }
  + Stack starts with the OS, then when the program is called main is pushed onto the stack, which holds the memory location for var. Then when we get to f(var) a activation record for f is pushed onto the stack on top of main. That record holds memory location for x and y, while passing var into varible x inside f. Sets y = x. then increments y. Then return goes to the bottom of the stack record and stores the return value. Then end of function occurs and the function is popped. Then we call G() which goes to the top of the stack where memory location n is stored. Then n gets passed from g to f creating a stack on top of G for F. Redoing the stack of f but with the new parameter n.