Note Title 9/11/2013

CS2100 - End of C++, + Simple linked lists

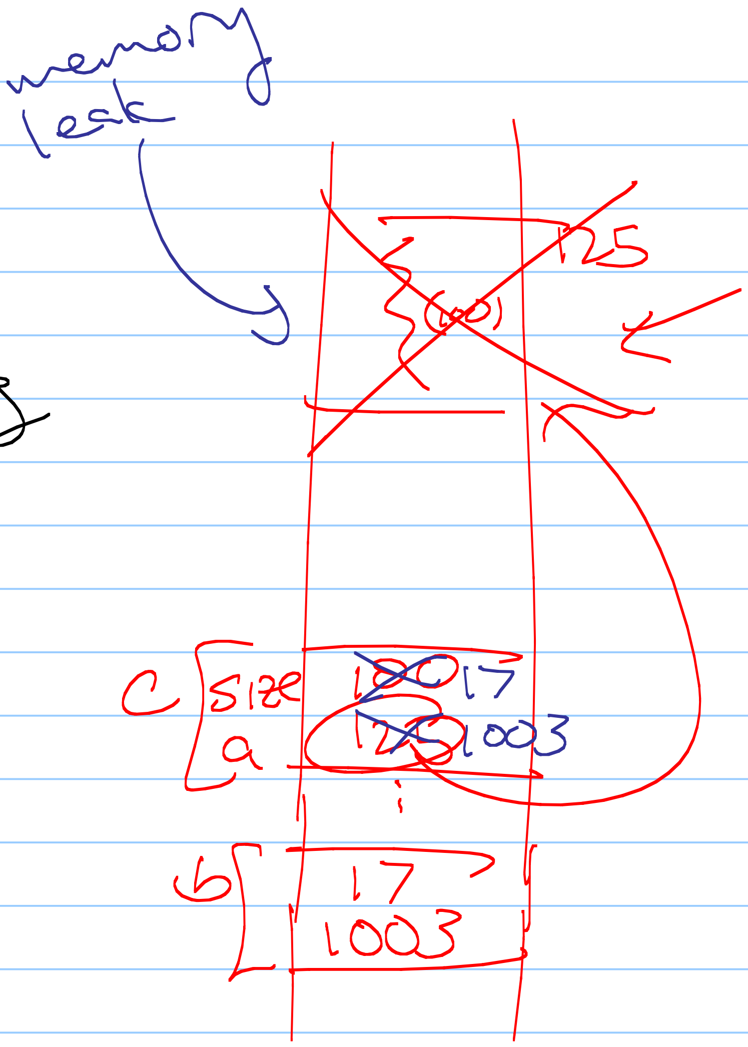
Announcements

Another issue:

```
MyFloatVec   c;
// use c for something
c = b;
```

What does this do?

BAD { - shallow copy
- memory leak



x=y=c;
x=x;

Solution: rewrite the "=" operation

```
MyFloatVec Operator=(const MyFloatVec& other) {
        if (this != &other) {
            size = other.size ;
            delete[] a;

            a = new float[size];
            for (int i=0; i<size; i++)
                a[i] = other.a[i];

      }
      return *this;
}
```

Recap: Housekeeping Functions

1. Copy Constructor
2. Destructor
3. Operator =

Why?

- memory leaks
- deep copies

Enum: user defined types

```
enum   Color   {RED , BLUE, GREEN };
                 0     1     2
enum   Rainbow {GOLD, SPARKLY };
Color  sky = BLUE;
Color  grass = GREEN;
Rainbow mine = SPARKLY;
                (1)
 if ( sky == BLUE )
      cout << "It's nice out today!" << endl;
```

NOT: `if(sky==mine)` <— error

Structs

useful for simple collections of ~~objects~~ data

Ex:
```
enum MealType {NO_PREF, VEG,
               REGULAR, KOSHER};

struct Passenger {
    string   name;
    MealType mealPref;
    bool     isFreqFlyer;
    string   freqFlyerNo;
}
```

Using structs

We can then create instances of a struct in the program:

```
Passenger pass = { "John Smith", VEG, true,
                   "1234" };

pass.mealPref = KOSHER;
```

ok: NOT objects

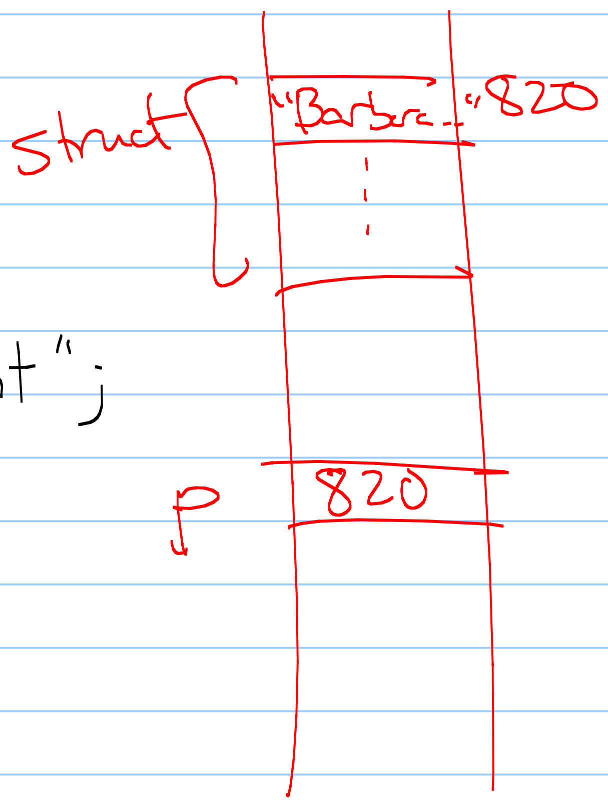
More Complex

```
Passenger* p;

p = new Passenger;

p -> name = "Barbara Wright";
p -> mealPref = REGULAR;

(*p).isFreqFlyer = false;
(*p).freqFlyerNo = "None";
```



Templates

If we want a function to work for multiple classes - eg int and floats - we can template the variable type.

Ex:

```
template <typename T>
T min(T a, T b) {
    if (a<b)
        return a;
    else
        return b;
}
```

Important:

Will work for any class with appropriate operators!

Ex:

```
int x = 53;
int y(96);

int z = min(x,y);

string a = "Hello";
string b = "Goodbye";

cout << min(a,b) << endl;
```

NOT

min(x, a)

or passenger

Templates in classes

These work in classes, also.

Important in data structures, so our
code will make a list of
ints or strings or lists!

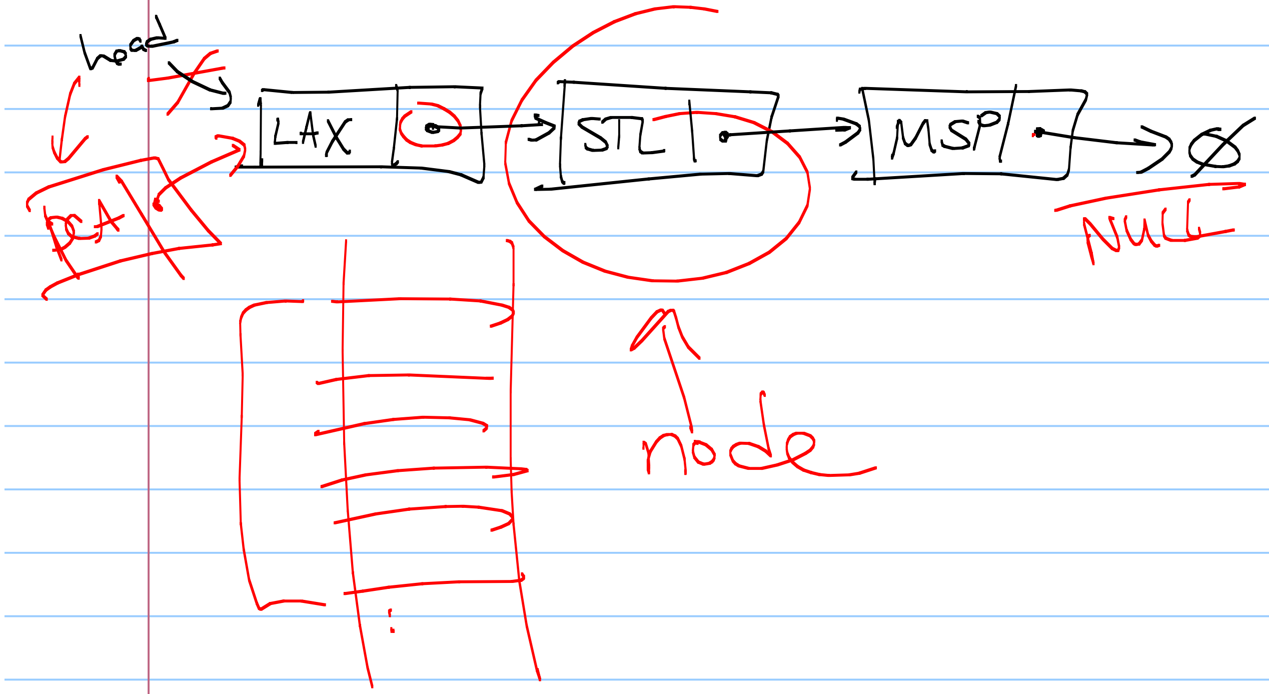
Using a template:

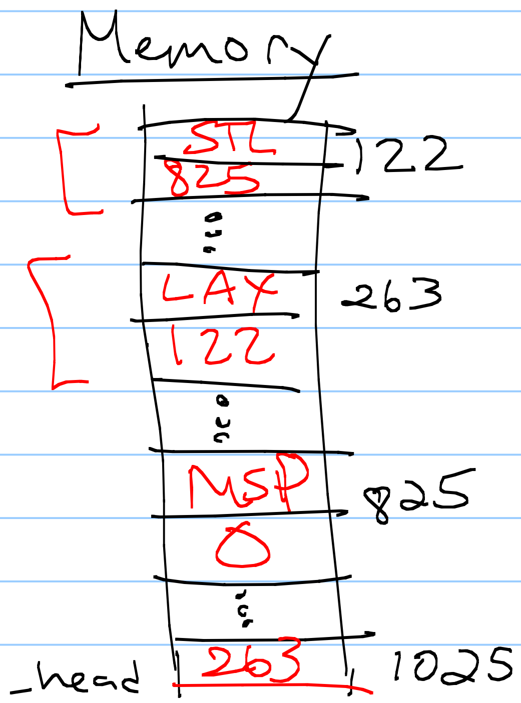
```
    MyList <int>     list1 ;

    MyList <string>  list2;
```

Singly Linked Lists

A collection of nodes that together form a linear ordering.





<u>Why this structure?</u>

<u>Note</u>: This is not the same as the list class which we'll write later.
(nor is it like Python lists)

This linked structure will show up in a lot of our data structures
- similar to arrays as a building block.

So why?

- more flexible: no max size, can expand easily

Implementation

What is a node & how do we code it?

data + pointer $\hookrightarrow$ struct SNode

Private data?

SNode $\bigcirc\!\!\!\!*$ head;
int size;

Functions?

insert
remove
test if empty

housekeeping!

Code

```
template <typename Object>
class SLinkedList {
    private:
        class SNode {
            private:
                Object _el ;
                SNode<Object>* _next;
        };
        SNode<Object>* _head;
```

Functions (listed in .h file)

```
public:

        SLinkedList();
       ~SLinkedList();
        bool  empty()    const;
        const  Object&  front()  const;
        void   addFront(const Object& e);
        void   removeFront();

  };
```

Next:

Let's code it!

(Will post .h + test file on
schedule page.)