Announcements

- Program 1 due next Friday (start this weekend!)

- First midterm in 2 weeks: either Wed., Sept. 23 or Thurs. Sept. 24 (with lab on other day) Preference?

- HW3 will come out next week, & program 2 will come out right before/after the midterm
Seems handy...

```java
int main(int a, int b)
{
    if (a > b)
        return a;
    else
        return b;
}
```

Consider a function:

```java
Template (Sec. 2.3)
```
Important: Will work for any class, as long as "a" has been defined!

```python
3
else

    return a

if (a < b):

    min (1, a, b) =

    
    template (templates)
```

This is a parameter list. Only one more called.

Function Templates:
Class Vector: a vector example

public:
    int capacity; // length of array
    Object* a; // array of elements

private:
    Class Basic Vector

    static void* clone (cloneable* a) { return new Basic Vector (a); }

    void* operator new (size_t s) { return new (this) Basic Vector (s); }

    void operator delete (void* p, size_t s) { free ((char*)p); }

    return object [r] { e } [::-1]; // access rth element

Obj + get elem [::-1] (int r) // access rth element

3

c = new Object [capacity]; // allocate changes

Basic Vector (int c = 10) 3 // constructor

public:
    int capacity; // length of array
    Object* a; // array of elements

private:
    Class Basic Vector

    static void* clone (cloneable* a) { return new Basic Vector (a); }

    void* operator new (size_t s) { return new (this) Basic Vector (s); }

    void operator delete (void* p, size_t s) { free ((char*)p); }

    return object [r] { e } [::-1]; // access rth element
(Sec. 1.1.3)

or just say pointer address.
So we can add to that number.
Then pointer is address of first element.
A just put a pointer to first element.
Can always set an array using new.
Make in C++, arrays are pointers!
```c
myvec[8] = 15

myvec. elem_at Rank(2). elem_at +4 Rank(8) = 15

vec_len = 5
vec = new int[5]

vec = (int *)malloc(sizeof(int) * 5);

vec[0] = 1;
vec[1] = 2;
vec[2] = 3;
vec[3] = 4;
vec[4] = 5;

free(vec);
```

Basic Vector + Basic Vector = Intvec

Back to BasicVector Usage
Exceptions are often in herited, gin
might have a set of similar ones...

Exceptions libraries doesn't always use.
Template libraries add to C++, as standalone.
Relatively new addition to C++, as standalone.

In C++ exceptions are "thrown", by code.

Exceptions
// Probably others to access message...

error_msg = error?

MathException (const string err)

public:

  String errorMsg;

private:

  Class MathException

Example: MathException
Class NegativeRootException (const string &err):
    public:
        NegativeRootException (const string &err):
            MathException (err) { }

Class NegativeDivideException (const string &err):
    public:
        NegativeDivideException (const string &err):
            MathException (err) { }

More specific exceptions:
will also catch NegaiveArrayBoundsException

```java
3
    catch (MathException e)
    { // handle any others
        catch (MathException e) { // handle division by 0
            catch (ZerodivideException e) { // handle division by 0 in Module X
                throw ZerodivideException("Divide by 0")
            }
            if (divisor == 0) { // some computations
                // try
                try { // catching Exception
                    // Throwing a Catching Exception
                } catch (Exception e) { // catch any other exceptions
                    // handle these exception
                }
            }
        }
    }
```
What happens if we had thrown a negative \( \text{Exception} \) if we had thrown a negative result excepción? Since that is closest matching catch, it would have been caught by the \text{MathException} catch block. In previous examples if we had thrown a program just aborts. If the exception is not caught, the

\[
\text{When divisor is equal to 0, it immediately}
\]
deallocating memory.

- May require cleanup, such as:

  - Often print error messages.
  - Depends on type of exception.

How do we recover?
Exceptions will be passed up - another means we don't have to handle Exceptions. We can handle appropriately so that can handle appropriately, let users know what to expect.

Exception. Also specify what exceptions might occur. When we declare a function, we should Exceptions in functions.
catch ErrorException —

my $calendar = Calendar::Gregorian();

my $calendar = Calendar::Gregorian();

Exception in calculation for any child class.

unless we can throw only these:

3 "function body"

3 \text{old calculator} \rightarrow \text{new (ReusableException\_Generator\_exception)}

Syntax: Exceptions in functions
What do these mean?

```c
void func1 () { //body

void func2 () throw () { //body
```
(See Sec. 2.5 and 4.1 for a review)

\texttt{on object} \downarrow

\texttt{a function that calls itself}

\texttt{What is it?}

\texttt{One final recap: Recursion}
Stack: a way to store a list of data

Ex: Web browser: Store history
last in is first out

Ex: Text editors: Store previously executed commands (LIFO)

undoes most recent action
The Stack Abstract Data Type (ADT) supports 2 main functions:

- push (x) - Insert object x at top of stack
- pop () - Remove top object from stack

To return top object from stack:

- push (3)
- push (11)
- push (5)
- push (13)

⇒ return 13

⇒ Stack (13)
(Like pop, but does not remove elt)

- pop: Returns object on stack without removing it
- empty(): returns true if stack is empty, false otherwise
- is_empty(): returns true if stack is

The stack has no objects in it.

Additional behaviors:
Debute it a create layer.
and if any is too small.
- maximum size to our short.
- keep track of "top", "element"
- private data on anyway.

I disagree.

I never said what this is.

Notice: