Review for Midterm II
• Monday, November 24, 11.00am-11.50am
• Try to show up at least 5 min early so we can start on time.

• The exam will focus on the material covered since Exam 1.
• Passing by reference vs passing by value
• Ch. 3: Random numbers
• Ch. 5: Classes, overloading
• Ch. 6: Vectors

• But in order for these topics to make any sense, you have to know the material in Ch. 1-5 (I/O, data types, arithmetic, if statements, loops, functions)
How to study for the exam

Read the slides.

Work through as many practice exams and problems as you can.

Look at your notes and examples your TA did.

Review homework assignments.

Read the book.
Syntax

Knowing the syntax also helps you write correct structure for your program.

Knowing the syntax reinforces other concepts.

If you are not confused about syntax the problem becomes clearer.

There is no excuse for not knowing the syntax. All it takes is some memorization.
Know the syntax for function prototype, function definition and for calling a function.

It is quite likely that when I ask you to write code that does some task I will ask you to write a function that does the task.

You should know what a void function is.

You should know how to write a function returning type bool and know how and when to use such a function.

You should know the difference between passing by value and passing by reference.
void foo (int &x, int &y, int z)
{
    cout << "x= " << x << " y= " << y << " z= " << z << "\n";
    x+=2;
    y=0;
    z % 2;
    cout << "x= " << x << " y= " << y << " z= " << z << "\n";
}

int main()
{
    int x=10;
    int y=20;
    int z=30;

    cout << "x= " << x << " y= " << y << " z= " << z << "\n";
    foo(z,y,x);
    cout << "x= " << x << " y= " << y << " z= " << z << "\n";
    if (x=10)
    {
        cout << "x is ten";
    }
    return 0;
}
Write a function called `alt_sum` that takes as an argument a vector of ints and returns an alternating sum of the vector. That is if the vector is

| 5 | 2 | 7 | 3 | 2 |

The alternating sum is: 5 -2 +7 -3 + 2 = 9. So the function should return 9.

```c++
int alt_sum(vector<int> numbers) {
    int sum = 0;
    for (int i=0; i < numbers.size(); ++i) {
        if (i%2==0)
            sum += numbers[i];
        else
            sum -= numbers[i];
    }
    return sum;
}
```
Just take the first line of the function definition and add a ; to the end.

```cpp
#include<vector>
using namespace std;
int alt_sum(vector<int> numbers);
```
Never put any types in function call!!

Function call only has names of variables.

cout << "The alternating sum is: " << alt_sum(numbers);
You should know the scope rules that we discussed in class.

What is a life span of a variable?

What does local scope mean?

More “inner” variables always supersede the “outer” variables.

Know what global variables are and how to declare them and when they are appropriate.

Understand what happens if you declare a variable inside a loop.

Review the “massive scope example”.
You should know how and why to seed a random number generator.

```c
srand((int)(time(0)));  
```

You should know how to get random numbers in a specified range.

The magic formula to get a number in a range $[a,b]$ is

```
a + \text{rand()} \% (b-a+1)
```

but you should know how to adapt this formula if need arises.

Ex: Write a single line of code that assigns to an `int x` a random even number in the range $[50,100]$.

```
// Get a number in range 25-50 then multiply by 2.
int x = 2 * (25 + \text{rand()} \% (50-25+1));
```
Write a function called random_vector that takes an argument of int \( n \) and returns a vector of size \( n \) filled with random decimals in the range 0.0 to 10.0 inclusive.

Solution:

```cpp
vector<double> random_vector(int n)
{
    vector<double>v;

    for (int i = 0; i < n; i++)
    {
        v.push_back( (double)(rand() % 101) / 10);
    }

    return v;
}
```
Random numbers

Assume that you have a vector<string> v that has been filled. Output a random element from this vector.
Classes

Know the syntax!

Declaring a class.

Implementing a class.

Know how to write a constructor, both default and regular.

Know how to overload an operator.
a) Write a class declaration for a class called Student, which has the following features: An object of class Student has two data fields string name and a vector of doubles called grades. There is a default constructor that sets name to a blank string and grades vector to an empty vector and a regular constructor. There is an accessor that returns the name of the student. There is an operator > that compares two students. This operator should compare the two students and return true if the current instance of Student has a higher average grades.

```cpp
class Student {
public:
    Student();
    Student(string new_name, vector<double> new_grades);
    string get_name() const;
    bool operator > (Student s);
private:
    string name;
    vector<double> grades;
};
```
Class implementation

Student::Student()
{
    name="";
    //Either do nothing to grades or:
    vector<double>blank;
    grades = blank;
}

Student::Student(string new_name,
vector<double>new_grades)
{
    name = new_name;
    grades=new_grades;
}
```cpp
string Student::get_name() const
{
    return name;
}

bool Student::operator>(Student s)
{
    double ave1, ave2, sum1=0, sum2=0;

    for (int i=0; i < grades.size(); i++)
    {
        sum1+=grades[i];
    }
    for (int i=0; i < s.grades.size(); i++)
    {
        sum2+=s.grades[i];
    }

    ave1 = sum1/grades.size();
    ave2 = sum2/s.grades.size();

    return ave1 > ave2;
}
```
int main()
{
    vector<double> g1, g2;
    string name1, name2;
    double grade;

    cout << "Name1 please: ";
    cin >> name1;
    cout << "Name2 please: ";
    cin >> name2;
    cout << "Enter a grade for " << name1 << ": ";
    while(cin >> grade)
    {
        g1.push_back(grade);
        cout << "Another grade: ";
    }
    cin.clear(); // These two lines just reset cin back to non-fail state
    cin.ignore(1000, '\n');
    cout << "Enter a grade for " << name2 << ": ";
    while(cin >> grade)
    {
        g2.push_back(grade);
        cout << "Another grade: ";
    }

    ...Continued on next slide...
Class implementation

Student s1(name1,g1);
Student s2(name2,g2);

if (s1 > s2)
    cout << s1.get_name() << " has better GPA than "
    << s2.get_name() << ".";
else
    cout << s2.get_name() << " has better GPA than "
    << s1.get_name() << ".";

return 0;
Vectors

You should know how to traverse through a vector.

You should also know how to go backwards through a vector.

We discussed a lot of important vector functions in class. You should be familiar with all of them.
Write a bool function that takes in a vector of ints a returns true if the entries are in increasing order and false otherwise.

```cpp
bool increasing(vector<int> v)
{
    int previous = v[0];
    for (int i=1; i < v.size(); i++)
    {
        if( previous >= v[i])
            return false;
        previous = v[i];
    }
    return true;
}
```
We want to search through a vector of integers and find the largest value occurring in the vector.

```cpp
int max(vector<int> v)
{
    int m = v[0];
    for(int j=1; j<v.size(); j++)
        if(m < v[j])
            m = v[j];
    return m;
}
```

It is important that we set m initially to some element in the vector. By default we pick the first element. Why can't we just set m to 0 initially?
Vectors

Write a function that takes in a vector of ints and an int and returns the number of times the int occurs in the vector.

```cpp
int frequency(vector<int> v, int value)
{
    int count=0;
    for (int i=0; i < v.size; i++)
    {
        if (value == v[i])
            ++count;
    }
    return count;
}
```
Vectors

```cpp
int number;
vector<int> numbers;

cout << "Please enter a number: ";

while (cin >> number)
{
    numbers.push_back(number);
    cout << "Please enter a number: ";
}

// Next two lines clear the cin buffer and reset cin to non-fail state
cin.clear();
cin.ignore(10000, '\n');

cout << "Enter a number to search: ";
cin >> number;

cout << number << " occurs " << frequency(numbers, number) << " times.";
```
Write a function called unique that takes as an argument a vector of ints called numbers and returns a vector of ints containing all the unique elements of numbers. For example, if numbers is:

```
2 10 4 2 2
```

The function should return the vector:

```
2 10 4
```
```cpp
#include <iostream>
#include <vector>
using namespace std;

vector<int> unique(vector<int> numbers)
{
    vector<int> v;
    bool found=false;
    for (int i=0; i < numbers.size(); i++)
    {
        for (int j=0; j < v.size(); j++)
        {
            if (v[j] == numbers[i])
                found=true;
        }
        if (!found)
            v.push_back(numbers[i]);
        found=false;
    }
    return v;
}
```
a) Write a function called \texttt{find\_smallest} that takes as an argument a vector and returns the INDEX of the smallest element in the vector.

Solution: You should know how to do this now!!! (It is almost identical to the max problem.) Note that we are returning the index not the value.

b) Write a function called \texttt{erase} that erases an element at index i.

\begin{verbatim}
void erase(vector<int> &v, int pos)
{
    for (int i = pos; i < v.size() - 1; i++)
        v[i] = v[i+1];
    v.pop_back();
}
\end{verbatim}
c) Write a function called `sort` that takes a vector `v` of ints and returns a sorted vector of ints in ascending order. You cannot use the sort function in the algorithm class. You must write this sort function by yourself.

For example, if the vector `v` is:

```
5  2  8  3  12  8
```

The returned vector should be:

```
2  3  5  8  8  12
```

HINT: Declare a vector `final` that will eventually hold all the sorted elements of `v`. This vector is the one you should return at the end. Utilize the functions you defined in a) and b).
Solution to c)

```cpp
vector <int> sort(vector<int>v)
{
    vector <int> final;
    int index=0;
    int size = v.size();
    for (int i=0; i < size; i++)
    {
        index=find_smallest(v);
        final.push_back(v[index]);
        erase(v,index);
    }

    return final;
}
```
Write a function definition of a function called `most_frequent`. This function takes as an argument a vector \( v \) of integers and returns the integer that appears the greatest number of times in the vector \( v \). For example if the vector \( v \) is:

\[
\begin{array}{cccc}
10 & 7 & 5 & 7 \\
\end{array}
\]

your function should return 7. If there is a tie for the most frequent element, you may return any most frequent element that is most convenient for you.
```cpp
int most_frequent(vector<int>v) {
    int max_freq=1;
    int max_elt=v[0];
    int freq=0;
    for (int i = 0; i < v.size(); i++) {
        for(int j=0; i < v.size(); j++) {
            if (v[i] == v[j])
                freq++;
        }
        if (freq > max_freq) {
            max_freq = freq;
            max_elt = v[i];
        }
        freq = 0;
    }
    return max_elt;
}
```