Final Review: Part I
Final exam

• The final exam is worth 30% of your grade, same weight as 2 midterms. Could be 50% if grading option 2 turns out better for you.

• Length is also roughly 2 midterms (~8 questions with parts).

• Very similar in style to 2nd midterm.

• Exam covers Ch. 1-7 + Sec 9.1, 14.4 and
How to study for the exam

Read the slides.

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

See what you missed on midterms.

Look at your notes and examples your TA did.

Review homework assignments.

Read the book.

Seems like a lot of work? IT IS! Get going.
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.
I/O

Know which way the arrows go with cout and cin!

To read a whole line of input into a string use getline(cin, my_string);

<iomanip> stuff
cout << fixed;
cout << setprecision(5) << number;
cout << setw(4) << col1 << setw(4) << col2;

You should know about cin buffer and fail state. Recall how we used cin in a while loop.

Good problem to check your knowledge: Homework 2.
Data types and arithmetic

Example: Casting / Integer division

```java
int a=5;
int b=7;
double result = (double)(a)/b;
```

The only unusual arithmetic operator is %. Be sure you understand it and know how to use it.

HW #1 should be crystal clear to you now.

Know all the shorthand notation for assignment and incrementation. +=, counter++, etc.

Good problem to check your knowledge: Midterm 1, problem 1
Remember to `#include <string>`

Concatination: Use a + to append one string to the end of another.

Memorize all the string member functions we talked about and make sure you understand how to use them.

Good problem to check your knowledge: Midterm 1, problem 5.
You should definitely know the syntax for all control structures.

You should be able to convert one type of loop to another.

You should know when to use a for loop and when to use a while loop.

You should be very comfortable with &&, || type expressions.

You should understand booleans.

Good homework to look at to review this stuff is the blackjack program.
Write a single line of code to write the following to the console:

```
The file path is:
"C:\system\new"
```

Translate the following description into code:

Assume you have an int variable N, which has been initialized. Perform the following algorithm on N. If N is odd, multiply it by 3 and add one. If N is even, divide it by two. This gives you a new value for N. Output the new value of N. Keep doing this process until N equals 1.
Typical short answer questions from old final

Solutions:

```cpp
cout << "The file path is:\n"C:\system\new"; 

while (N != 1) 
{ 
  if (N % 2 == 0) 
  { 
    N/=2; 
  } 
  else 
  { 
    N*=3; 
    N++; 
  } 
  cout << N << endl; 
} 
```
a) Assume an int n has been initialized. Write a snippet of code to output to the console its digits vertically. If n is 104533, the result should be:

1
0
4
5
3
3

Hint: Use int division and % operator to your advantage.

b) You are given a connection to a file through ifstream object fin. Write a snippet of code that counts the number of spaces in the file.
Functions

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.

Work through some examples where you chase values of variables like we did in class.
Write a function called divide_by_2 that takes as an argument a vector of ints and returns a vector where all even elements are divided by 2. If the vector is:

| 5 | 2 | 8 | 3 | 12 |

The returned vector should be:

| 5 | 1 | 4 | 3 | 6 |

```cpp
vector<int> divide_by_2(vector<int> v) {
    for(int i=0; i<v.size(); i++) {
        if (v[i] % 2 == 0) {
            v[i] /= 2;
        }
    }
    return v;
}
```
Just take the first line of the function definition and add a ; to the end.

```cpp
#include <vector>
using namespace std;
vector<int> divide_by_2(vector<int> v);
```
Never put any types in function call!!

Function call only has names of variables.

vector<int> initial;

// Somehow we fill initial vector

vector<int> final = divide_by_2(initial);
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 thoroughly understand the difference between the two.

You should also know when to pass by reference:

• Passing a class to a function

• Passing a vector to a function

Example: Swap function

```c
void swap(int &a, int &b)
{
    int temp;
    temp = a;
    a=b;
    a=temp;
    b=temp;
}
```
Search through a vector and return the first position where value occurs.

```cpp
int find(const vector<int> &list, int value)
{
    int i = 0;
    while (i < list.size() && list[i] != value)
    {
        i++;
    }
    return i;
}
```

Here we use passing by reference for efficiency
Search through a vector and return the first position where value occurs.

```c++
void foo(Employee &e)
{
    e.set_salary(1500);
}
```

If we want the change to `e` be recorded in `main` we must pass `e` by reference.
Random numbers

You should know how and why to seed a random number generator.

`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.

Good example of this is the question: Write a single line of code that assigns to an `int x` a random even number in the range \([50,100]\).

```c
// Get a number in range 25-50 then multiply by 2.
int x = 2 * (rand() % (50-25+1) + 25);
```
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 class called Team has the following data: name, wins, losses. name is type string and wins and losses are type int. The class has a default constructor which sets name to “” and wins and losses to 0 and a regular constructor that takes as arguments a name, wins and losses. It has an accessor get_name that returns the name. It also has a mutator called defeats that takes as an argument another Team object and when called, it increments wins by one for the instance object and increments losses for the argument object by one. It has an operator > that compares the ratio of wins to losses.

An example of the use of class is:

Team A("Isotopes", 2, 14);
Team B("Shelbyvillians", 6, 5);

B.defeats(A); //B's wins are incremented and A's losses are incremented.

if (A > B)
    cout << A.get_name() << " is the better team."
else
    cout << B.get_name() << " is the better team."
class Team
{
private:
    Team();
    Team(string new_name, int new_wins, int new_losses);
    void defeats(Team &T);
    bool operator > (Team T) const;
    string get_name() const;
    int get_wins() const;
    int get_losses() const;
private:
    string name;
    int wins;
    int losses;
};
Team::Team()
{
    name="";
    wins = 0;
    losses = 0;
}

Team::Team(string new_name, int new_wins, int new_losses)
{
    name = new_name;
    wins = new_wins;
    losses = new_losses;
}
```cpp
void Team::defeats(Team &T)
{
    wins++;  
    T.losses++;  
}

bool Team::operator > (Team T) const
{
    if (losses==0)
        return true;
    if (T.losses == 0)
        return false;
    if ((double)wins/losses > (double) T.wins / T.losses)
        return true;
    else
        return false;
}

string Team::get_name() const
{
    return name;
}
```