CISC 3110   Midterm Review

Rough breakdown of topics:

20%   Base Linux usage (basic comamnds, emacs/vi usage)
20%   g++, make (Unit 1)
30%   Pointers and dynamic allocation (Unit 2)
30%   Classes, Objects, and Object-Oriented Programming (Unit 3)

Study advice:
   • Do exercises in the book (or from other readings).
   • Review "Application Activities" questions, below
   • Review your solutions to the programming projects.

Note: you should expect most of the exam questions to be very familiar. I will use a lot of the
questions from the RATs and the application activities, though I may make some small changes or ask
them in a different way. You should expect the midterm, like the application activities, to be a mix of
different kinds of questions—multiple choice, short answer, code analysis, code writing….

1. Which of the following "understand" C++?

   a) preprocessor
   b) compiler
   c) linker
   d) both a and b
   e) both b and c

2. Which is the **<u>best</u>** sequence of commands to use to compile and run a program from the source files `source1.cc` and `source2.cc`? Both files also `#include` the header file `header.h`.

a)
```
$ g++ -Wall source1.cc
$ g++ -Wall source2.cc
$ ./a.out
```

b)
```
$ g++ -Wall -c source1.cc
$ g++ -Wall -c source2.cc
$ ./a.out
```

c)
```
$ g++ -Wall source1.cc source2.cc -o runme
$ ./runme
```

d)
```
$ g++ -Wall -c source1.cc
$ g++ -Wall -c source2.cc
$ g++ source1.o source2.o -o runme
$ ./runme
```

e)
```
$ g++ -Wall -c source1.cc header.h
$ g++ -Wall -c source2.cc header.h
$ g++ source1.o source2.o -o runme
$ ./runme
```

3. Assuming `helloworld.cc` is a valid C++ program (no syntax errors, etc), which sequence of commands will NOT produce a valid executable?

a)

```
$ g++ -E helloworld.cc > helloworld.ii
$ g++ -Wall -S helloworld.ii
$ g++ helloworld.s -c
$ g++ helloworld.o
```

b)

```
$ g++ -Wall -S helloworld.cc
$ g++ helloworld.s
```

c)

```
$ gcc helloworld.cc
```

d)

```
$ g++ helloworld.cc -c
$ g++ helloworld.o
```

e)

Neither a nor c will produce a valid executable.

4. Consider the 4 snippets of text (**1**, **2**, **3**, and **4**). These are output fragments from various phases of the compilation of `helloworld.cc`.

```
main:                                              1
.LFB971:
        .cfi_startproc
        pushl   %ebp
        .cfi_def_cfa_offset 8
        .cfi_offset 5, -8
        movl    %esp, %ebp
        .cfi_def_cfa_register 5
        andl    $-16, %esp
        subl    $16, %esp
```

```
□                                                  2
C9><C3>Hello,
world.^@^@^@^@s^@^@^@^@GCC: (Ubuntu
4.8.4-2ubuntu1~14.04)
4.8.4^@^@^@^@^T^@^@^@^@^@^@^@^@^AzR^@
^A^AESC^L^D^D<88>^A^@^@^\^@^@^@^\^
@^@^@^@^@^@^@4^@^@^@^@A^N^H<85>
^BB^M^Ep<C5>^L^D^D^@^@^\^@^@^@<^@^
@^@4^@^@^@?
```

```
#□1 "helloworld.cc"                                3
# 1 "<built-in>"
# 1 "<command-line>"
# 1 "/usr/include/stdc-predef.h" 1 3 4
# 1 "<command-line>" 2
# 1 "helloworld.cc"
# 1 "/usr/include/c++/4.8/iostream" 1 3
# 36 "/usr/include/c++/4.8/iostream" 3
```

```
□                                                  4

^@^@^@<A0><85>^D^H^@^@^@^@^R^@^@
^@<FE>^@^@^@p<85>^D^H^@^@^@^@^R^
@^@^@<A4>^^@^@^@@<A0>^D^H<8C>^^@^@
^@^Q^@^Y^@^@libstdc+
+.so.6^@__gmon_start__^@_Jv_RegisterClasses^
@_ITM_deregisterTMCloneTable^@_ITM_register
TMCloneTable^@_ZSt4endlIcSt11char_traitsIcEER
St13basic_ostreamIT_T0_ES6_^@_ZSt4cout^@_Z
StlsISt11char_traitsIcEERSt13basic_ostreamIcT_ES
5_PKc^@_ZNSt8ios_base4InitC1Ev^@_ZNSt8ios_
base4InitD1Ev^@_ZNSolsEPFRSoS_E^@libc.so.6^
@_IO_stdin_used^@__cxa_atexit^@__libc_start_
main^@GLIBCXX_3.4^@GLIBC_2.0^@GLIBC_2.1
.3
```

Which sequence represents
<output of preprocessor>, <output of compiler>, <output of assembler>,
<output of linker>?

  a) 1, 2, 3, 4
  b) 1, 3, 2, 4
  c) 3, 1, 2, 4
  d) 2, 1, 3, 4
  e) 3, 2, 4, 1

6. Which lists the components of the compilation process in order from MOST likely to generate an error to LEAST likely to generate an error?

a) Compiler, assembler, preprocessor, linker
b) Compiler, assembler, linker, preprocessor
c) Compiler, linker, preprocessor, assembler
d) Linker, compiler, assembler, preprocessor
e) Preprocessor, compiler, assembler, linker

7. Which of these is true?

a) A C++ statement like `int myfun(char c, int n);` is called a *function declaration.* It tells the compiler that the full implementation—the *function definition*—is coming later, perhaps from another file.

b) A C++ statement like `int myfun(char c, int n);` is called a *function definition.* It tells the compiler that the full implementation—the *function declaration*—is coming later, perhaps from another file.

c) A C++ statement like `int myfun(char c, int n);` will cause a linker error if there is no full function implementation.

d) Both a and c

e) Both b and c

8. Consider the following file `things.h`

```
#ifndef THINGS_H
#define THINGS_H

#include<string>

int thing1(int x, int y);
std::string thing2 (std::string x, std::string y);

#endif
```

Which of the following is true?

a) during the g++ compilation process, the preprocessor notes that the symbol `THINGS_H` has been defined.
b) `things.h` should be passed to g++ on the command line
c) `things.h` should be `#include`-d in other C++ source files
d) Both a and c
e) Both b and c

9. Consider the following file `otherthings.h`

```
#include<string>

class Things {
      public:
              int thing1(int x, int y);
              std::string thing2(int x, int y);
};
```

(Something is missing, right?)

And also the file `lotsofwork.h`

```
#ifndef LOTSOFWORK_H
#define LOTSOFWORK_H

#include "otherthings.h"

int foo();

#endif
```

Assume these are accompanied by `.cc` files that define the necessary functions.

Your task: write a `main.cc` that demonstrates the problem caused by missing include guards. Also write a brief description of how this problem will manifest (will the compiler complain? Will the program fail to run? What error message will you see, if any?)

11. Which of these statements is true?

    a) `make` can only execute compiler commands.
    b) A `make` rule consists of targets, prerequisites, and recipes.
    c) Every `make` target requires an explicit recipe.
    d) By default, `make` starts with the target with the most prerequisites.
    e) Both b and c.

12. Write a makefile for the project described below.

The file `shapes.cc` includes definitions relating to drawing primitive graphics on the screen; the publicly useful classes and functions are declared in `shapes.h`.

The file `textures.cc` includes definitions relating to rendering of textures on the screen; the publicly useful classes and functions are declared in `textures.h`.

The file `sprites.cc` includes definitions relating to the animation of graphical "sprites"; the publicly useful classes and functions are declared in `sprites.h`. Some of the classes and functions in `sprites` rely on classes and functions from `shapes` and `textures`.

The main program, `main.cc`, uses `sprites`, `textures`, and `shapes` to display a simple animation.

13. Modular design of software

   a) Has finally become possible with recent innovations in compiler design.
   b) Helps to make more maintainable software
   c) Makes it easier for teams to write software efficiently
   d) Both a and b
   e) Both b and c

14. How should Alice, Bernardo, and Cong divide the work so that they can each be as independent as possible? Remember:

> After analyzing the intended behavior of this application, the three decide that they need to implement 5 functions: getint, putint, newline, sum, and product—as well as main.

15. What do Alice, Bernardo, and Cong need to agree on before they can start working individually?

(They already agree on the purpose of the program, of course.)

16. How can they best use g++ to support their work? Can they run the application when they're working individually? If not, how can they make sure their work is as free from problems as possible?

17. But wait: if we split up a project using modular design, it seems like we'll have this problem:

a) You have to get everyone's code together before the compiler can find the syntax errors.
b) You have to get everyone's code together before you can run it to see whether there are any bugs.
c) Alice can't use Bernardo's module in her code, because she has no way of knowing what's in Bernardo's module.
d) Cong forgets everything after 10 minutes, so his implementation isn't going to match the interface Alice and Bernardo are expecting him to provide.
e) If Bernardo realizes there's a more efficient way to implement his functions, it's too late, because changing his code might cause problems for Alice and Cong.

1. What is the output of this code fragment?

```
int x=3, y=4, z=5;
int *ptr1, *ptr2, *ptr3;

ptr1 = &x;
ptr2 = &z;
ptr3 = &y;

*ptr1 = *ptr2;
ptr3 = ptr2;
*ptr2 = *ptr2 + 10;

std::cout
        << std::setw(3) << x
        << std::setw(3) << y
        << std::setw(3) << z
        << std::endl;
```

    a)   3 14 5
    b)   5 4 15
    c)   5 4 4
    d)   5 14 5
    e)   3 15 15

2. What will happen when we try to compile and run this program?

```
1 #include <iostream>
2 using namespace std;

3 int main() {

4    char c = '?';
5    int x=10, y=20;

6    int *ptr1;

7    ptr1 = &x;
8    cout << *ptr1 << " ";
9    ptr1 = ptr1 - 1;
10   cout << *ptr1 << endl;
11 }
```

a) The compiler complains about an illegal assignment on line 7.
b) The compiler complains about an illegal dereference on line 8.
c) The compiler complains about an illegal dereference on line 9.
d) The program compiles, but crashes with a "Segmentation fault." error.
e) The program compiles and runs, but the output is 10 1063089092

3. What will happen when we try to compile and run this program?

```
 1 #include <iostream>
 2 using namespace std;

 3 int main() {

 4     char c = '?';
 5     int arr[3] = {4, 5, 6};

 6     int *ptr1;

 7     ptr1 = arr;
 8     cout << *ptr1 << " ";
 9     ptr1 = ptr1 + 1;
10     cout << *ptr1 << endl;
11 }
```

a) The compiler complains an illegal initialization on line 5.
b) The compiler complains about an illegal assignment on line 7.
c) The compiler complains about an illegal dereference on line 10.
d) The program compiles and runs, but the output is 4 7468992846
e) The program runs with output 4 5

4. Assuming we have the following declarations:

```
const int SIZE = 100;
int array[SIZE];
int sum=0, *ptr;
```

Rewrite the following code

```
for (int i=0; i < SIZE; i++) {
    sum += array[i];
}
```

with these restrictions:

1. You must use a `while` loop rather a `for` loop
2. You may *not* declare any additional variables.
3. You may *not* use any square brackets `[]`

5. Determine the output of the following program. Draw and update a diagram of memory to support your work. Write your solution (the output) and your diagram on the board.

```cpp
#include <iostream>
#include <iomanip>

const int SIZE = 10;

int main() {

    int arr[SIZE];
    int *ptr1, *ptr2;

    ptr1 = arr;
    while (ptr1 < arr+SIZE) {
        *ptr1 = ptr1-arr;
        ptr1++;
    }

    ptr1 = arr;
    ptr2 = arr+SIZE-1;
    while (ptr2 > ptr1) {
        if (*ptr2 > *ptr1) {
            *ptr2 = *ptr1;
            *ptr1 = *ptr2;
        }
        ptr1++;
        ptr2--;
    }

    for (int i=0; i<SIZE; i++) {
        std::cout << std::setw(3) << arr[i];
    }
    std::cout << std::endl;
}
```

6. What's wrong with this implementation of swap() ?

```
void swap(int* x, int* y) {

        int *temp;

        temp = *x;
        *x = *y;
        *y = temp;
}
```

a) Nothing
b) The compiler will issue an error because `int*` values aren't compatible with `int` values.
c) It will crash because `temp` doesn't point at anything.
d) It will run but it won't swap any values.
e) It will run but it just copies the first value into the second instead of swapping.

7. The "dynamic" in dynamic memory allocation means that:

a) Programs that have frequently-changing data should store it in special "dynamic memory."
b) The amount of memory allocated to a program may change from one run to the next.
c) The amount of memory allocated to a program may change during one run.
d) Both a and b
e) Both b and c

8. What's the problem with this code fragment?

```
2 int size, *ptr;

3 std::cout << "How many? ";
4 std::cin >> size;

5 if (size > 0) {
6     int *array = new int[size];
7     int sum = 0;
8     for (int i=0; i < size; i++) {
9         std::cin >> array[i];
10    }
11   ptr = array;
12    for (ptr=array+size-1; ptr > array; ptr--) {
13        std::cout << std::setw(3) << *ptr;
14    }

15    std::cout << std::endl;
16    delete[] array;
17 }
```

    a) Nothing.
    b) ptr points at an "illegal" addresses.
    c) The array isn't declared as an array.
    d) Dangling pointer.
    e) Memory leak.

9. Deleting line 16

    a) Has no effect.
    b) Solves the problem.
    c) Turns the problem into a memory leak.
    d) Causes the program to crash.

10. The problem with this code

```
1 int *ptr1 = new int[100];
2 int *ptr2 = new int[200];
3 ptr1 = ptr2;
4 ptr2 = ptr1;
5 delete[] ptr1;
6 ptr1 = NULL;
7 delete[] ptr2;
```

is

    a) Memory leak
    b) Dangling Pointer
    c) Crash
    d) All of the above
    e) None of the above

11. Write a function that takes an array of strings and returns an array containing the lengths of each nonempty string (empty strings should be ignored). Its signature is

```
int *getLengths(string[] array, int arraysize);
```

(Remember that you can get the length of a string s by saying s.length() and you can find out if it's empty with s.empty().)

12. Code like this

```
3 std::cout << "How many? ";
4 std::cin >> size;
5 if (size > 0) {
6    int *array = new int[size];
```

is great when the user knows ahead of time how much data she's going to give the program. But sometimes no-one knows in advance how much storage is necessary—what if you need storage for each user who joins a chat room? Or for each tweet from the people you follow? These things all happen in real time. What's the best way to deal with this kind of very unpredictable data?

    a) Ask the user how much storage to use. If more is needed, just allocate twice as much space and copy everything over.
    b) As the programmer, we should just pick a size limit and not even bother to ask the user.
    c) As the programmer, just pick a size limit that's way bigger than could ever be needed.
    d) Use a data structure that is designed to grow dynamically, just add more space as needed.
    e) Use Java; C++ can't handle that kind of unpredictability.

1. Suppose we were trying to define an abstract data type for the Tic-Tac-Math board. Which of these would *not* belong to such a definition?

   a) A Tic-Tac-Math board is a collection of 9 values, each between 1 and 9.
   b) Empty cells on the board are represented as zeros.
   c) A Tic-Tac-Math board is *winning* if three values from a "row," "column", or "diagonal" sum to 15.
   d) Each value can only appear on the board once.

2. Consider the following abstract data type for a Deck of cards:

A card must have one of four suits: hearts, diamonds, clubs, spades
A card must have one of 13 values: two-ten, jack, queen, king, ace.

A Deck *contains* 52 cards
In a Deck that has been *shuffled*, cards are removed in random order
You can *deal* the top card in a Deck by removing it

What's the problem:

a) This doesn't include any operations.
b) This doesn't fully specify the values.
c) This doesn't provide any rules of the game.
d) This DOES provide implementation details

3. Here's the `Circle` class from the book (the version from Figure 7.1, p 389):

```
class Circle {
   public:
      double radius = 0.0;

      void setRadius(double r) { radius = r; }
      double getArea() { return 3.14 * pow(radius, 2); }
};
```

Suppose we then declare a `Circle` object:

```
Circle roundThing;
```

Which of the following statements best illustrates the problem with this `Circle` definition?

a) `roundThing.setRadius(47);`
b) `std::cout << roundThing.radius;`
c) `double area = roundThing.getArea();`
d) `roundThing.radius = 0.5;`
e) `delete roundThing;`

4. The next version of `Circle` declares `radius` as a `private double`. What's the best implementation of `setRadius()`? (Assume any needed header files are `#include`-d)

a)
```cpp
void Circle::setRadius(double r) { radius = r; }
```

b)
```cpp
void Circle::setRadius() {
      std::cout << "Please enter the circle's radius ";
      std::cin >> radius;
      std::cout << std::endl;
}
```

c)
```cpp
bool Circle::setRadius(double r) {
      if (r > 0.0) {
            radius = r;
            return true;
      } else {
            radius = 0;
            return false;
      }
}
```

d)
```cpp
void Circle::setRadius(double r) {
      radius = r;
      std::cout << "Radius set; new area is " << getArea(); << std::endl;
}
```

e) `void Circle::setRadius() { radius = sqrt(getArea()/3.14); }`

5. Write a complete class called `Date`. Its public members must include these methods (return types are up to you):

```
setDate(int day, int month, int year);
getDay();
getMonth();
getYear();
```

Which is the best default constructor for the `Date` class? (Assume `day`, `month`, `year` are declared as `int` members.)

a) It would be best not to write one at all.

b) `Date::Date() { day = month = year = 0; }`

c) `Date::Date() { day = month = 1; year = 1970; }`

d)
```
Date::Date() {
    std::cout << "Enter values for day, month, and year, separated by space: ";
    std::cin >> day >> month >> year;
}
```

Which is the best destructor for the `Date` class? (Assume `day`, `month`, `year` are declared as `int` members.)

a) It would be best not to write one at all.

b) `Date::~Date() { day = month = year = 0; }`

c) `Date::~Date() { delete day; delete month; delete year; }`

d) `Date::~Date() { std::cout << "Welcome to the end of time." << std::endl; }`

Which of these sets of constructor declarations is illegal?

a)

```
Rectangle();
Rectangle(int width);
Rectangle(int length, int width);
```

b)

```
Rectangle(int width = 1, int length = 1);
Rectangle(std::string name, int width = 1, int length = 1);
```

c)

```
Rectangle(int width = 1);
Rectangle(int perimeter, int width = 37);
```

d)

```
Rectangle(int width, int area);
Rectangle(int width, int length, std::string name);
```

e) None of the above.

A private class member function can be called by

a) any other function
b) any other function in the same object
c) only public functions in the same class
d) only private functions in the same class
e) any function in the same class

Here is the `SimpleStat` declaration from the book:

```
class SimpleStat {
      private:
             int largest;
             int sum;
             int count;

             bool isNewLargest(int);

      public:
             SimpleStat();
             bool addNumber(int);
             double getAverage();
             int getLargest() { return largest; }
             int getCount() { return count; }
};
```

What's the worst thing that would happen if `isNewLargest()` were declared as `public`?

a) Whoever's using the class would get too much access to class/instance data.
b) Whoever's using the class could be confused about using `isNewLargest()` vs `addNumber()`
c) The implementation of `getLargest()` would no longer work properly.
d) The compiler would complain about "Illegal public function."
e) It makes no difference at all.

Reference parameters are so last semester. But *constant* reference parameters are all the rage. What problem do constant reference parameters solve?

a) It can be expensive to copy the objects we're passing to a function.
b) It's risky to have two ways to access a single object.
c) If you want to avoid copying costs, you shouldn't have to risk having your data changed.
d) The compiler can't tell when a function is changing an object's data.
e) I'm tired of people constantly referring to Kanye.

So, if constant reference parameters "protect an object when it is passed as an argument," what kind of protection are we talking about? Specifically, suppose Alice has implemented a class called `Hammertime`, and Bob is implementing a function `void stop (const Hammertime& h)`. Which of these correctly (best) describes the situation:

a) Alice is protecting against Bob (or anyone else, for that matter) improperly changing `h`.
b) Alice is protecting against Bob (or anyone else, for that matter), using `Hammertime` in a way she doesn't like.
c) Bob is protecting against bugs in Alice's code messing up his function's behavior.
d) Bob is protecting against his own mistakes changing `h` when he doesn't want to.
e) Bob is protecting against someone using `stop()` to make changes to `h`.

In addition to protection with minimum copying, do constant reference parameters offer any other benefits?

a) Nope, not so much.
b) They reduce the chance of memory leaks.
c) They let the compiler know it may be able to optimize the function.
d) They provide a visible guarantee to other programmers that values won't be changed.
e) They're chock full of the vitamins healthy objects need.

Suppose we have the class `Polygon` and the class `Point` defined:

```
class Polygon {                           class Point {
   public:                                   public:
      // several groovy constructors             // several groovy constructors
      Point getCenter();                         double getX();
      double getArea();                          double getY();
      int getNumPoints();                 // move the Point right xDelta and up
      void shift(double xDelta,           yDelta
                 double yDelta);                 void shift(double xDelta,
      void rotate(double degrees);                          double yDelta);
      bool intersects(Polygon p);
                                            private:
   private:                                     double x;
      Point* corners;  // array of points       double y;
      int numPoints;                       };
};
```

Write

```
Point(double x, double y);
Point();
Polygon(Point[] arr, int numPoints);
```

Also, write any necessary destructors

Write the `Polygon::shift()` method. This method shifts the `Polygon` xDelta units to the right and yDelta units up.

Consider the `Polygon::intersects()` method. How would this method access the points that make up p, the `Polygon` parameter?

a) We need to write a getter method like `Point* getPoints()` for the `Polygon` class.
b) `p.corners[0]` will give us the first `Point` of p.
c) It's not necessary; we could just call `p.intersects()` to do the actual computation.
d) It's not possible; the points are declared `private` so they can't be accessed by another object.