

1 Background

The questions in this study guide are designed to be difficult and help you identify knowledge gaps that warrant further study. By request, this study guide is compiled to help you prepare for final exams. Instruction during the final week of lecture may or may not cover sections of this packet. Please use this as you see fit. We hope this helps you prepare for your CSCI 1300 final exam and future studies. This packet does its best to cover all topics of the course, but is not exhaustive. Supplement with additional material as you see fit.

We strongly recommend reviewing the following sections of your textbook "Brief C++ Late Objects" - Cay Horstmann, including the embedded "Self Check"s:

- Chapter 1 subsections: 2 - 4
- Chapter 2 subsections: 1 - 3, 5
- Chapter 3 subsections: 1 - 7
- Chapter 4 subsections: 1 - 3, 5, 6, 8 - 10
- Chapter 5 subsections: 1 - 10
- Chapter 6 subsections: 1 - 7
- Chapter 7 subsections: 3, 7
- Chapter 8 subsections: 1 - 4
- Chapter 9 subsections: 1 - 6, 9, 11

Some solutions will be provided in a solution document.

SOLUTION: Solutions code available [here](#). Additional solutions in red in-situ.

If you cannot think of a problem worth solving in the self-study section: ask a peer, ask a member of the course staff (in office hours or on EdStem), ask an AI that you trust.

In general, you should think about the following tasks for any given self-study.

1. Write out what you know about the topic. Identify what is tricky about this topic.
2. Define a problem to solve: draw a cartoon of the problem, define some expected input and result, confirm that the defined problem will address whatever is tricky about the topic.
3. After you have completed the work on paper, validate your work using your preferred coding environment. Prioritize paper studies first.
4. Decide if you need to study this further and construct additional study materials.

2 Compilation

1. Compilation

- (a) What is the shortest command used to compile a file named `driver.cpp`?

SOLUTION: `g++ driver.cpp`

- (b) What if `driver.cpp` has the line `#include "className.h"`, following best practices for naming conventions, how do you correctly compile the driver file?

SOLUTION: `g++ className.cpp driver.cpp`

observe that the compilation uses `className.cpp`, NOT `className.h`

- (c) What flags should you also consider using? What is the purpose of each compilation flag?

SOLUTION:

- `-std=c++17`: set the c++ standard to use [here 2017] (note that there are new standards every 3 years)
- `-o whatever.o`: specify the name of the executable [here `whatever.o`], rather than `a.out` for Linux/Mac or `a.exe` for Windows
- `-Wall -Werror -Wpedantic -Wsign-compare`: each of these are well detailed in this semesters workbook

- (d) Given that you have compiled the code, how do you execute the code from the command line?

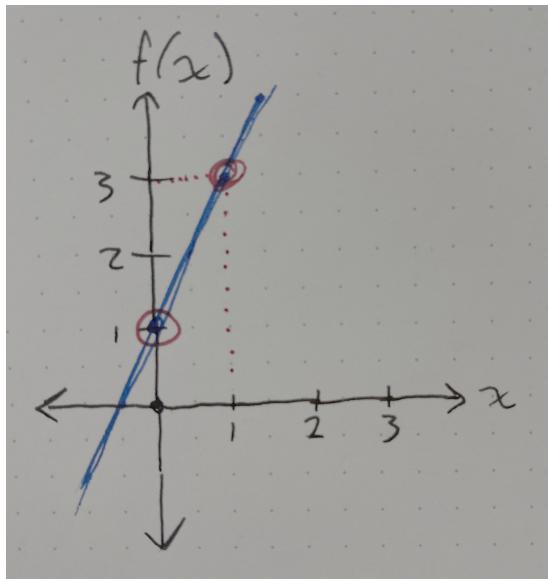
SOLUTION:

- LINUX/MAC: `./a.out`
- Windows: `.`
`a.exe`
- custom: using the `-o` flag, you may specify a different file

2. Essentials:: For each of the following topics, in your own words: (1) define the principles of the topic (2) give one example of the topic (3) identify and explain any important nuances about the topic (4) decide for yourself if you need to study this particular topic further and seek resources for it (textbook, workbook, lecture documents, lecture videos, peers, EdStem, course staff, youtube, the open internet, and more). Use additional paper as needed.
- (a) essential types and their operations: bool, short, int, float, double, char, string
 - (b) conditions (if/else-if/else) and switches
 - (c) functions (pass-by-value vs pass-by-reference)
 - (d) loops
 - (e) arrays
 - (f) streams (input vs output) (`getline` vs `>>`)
 - (g) structs/instances
 - (h) classes/objects
 - (i) vectors
 - (j) sorting
 - (k) recursion

SOLUTION: No solutions provided for this sample, it is up to you to verify your understanding of the topic using the resources that are available to you. Ask your peers, use office hours, use EdStem.

3. Writing Functions: Warm Up:: The questions on this page ask about the image provided (a function in graph form):



- (a) What is the expected value of $f(0)$?
SOLUTION: 1, easy to see on the $f(x)$ axis
- (b) What is the expected value of $f(f(0))$?
SOLUTION: 3, $f(0)$ equals 1, so it must be the case that $f(f(0)) = f(1)$. $f(1)$ is shown with dashed lines on the graph, value is 3
- (c) Is $f(f(0))$ a recursive call?
SOLUTION: No. This is a nested call. Those are different topics. You have had lots of experience with nested calls by this point in the semester.
- (d) What is the best representative type for x and $f(x)$ if we wanted to write code about this?
SOLUTION: `double` or `float` would be preferred. The graph is continuous and includes values like π . If we used an `int` then we wouldn't have a line, we would just have dots at each relevant whole number.

- (e) Author code for the pictured function. Use a function with a reasonable name and parameters as needed. Do not use `cin`:

SOLUTION: see below in black.

```
double f( double x )
{
    return 2 * x + 1;
}
```

- (f) What library is used for testing? How do we include it in our code?

SOLUTION: `#include<cassert>`

- (g) Author assertions for $f(0)$ and $f(f(0))$

SOLUTION: There are so many ways to do this. I'll keep the tight solutions here.

```
assert( f(0) == 1 );
assert( f(f(0)) == 3 );
```

- (h) Now that you've done this on paper, go check your work in a coding environment of your choosing.

SOLUTION: No solutions provided for this sample, it is up to you to verify your understanding of the topic using the resources that are available to you. Ask your peers, use office hours, use EdStem.

4. Reading Functions: guided:: The questions on this page ask about the recursive g function defined below:

```
string g( string x )
{
    string y = "";
    switch ( x.length( ) % 3 ) {
        case 0:
            y += "Alpha";
            break;
        case 1:
            y += "Bravo";
            break;
        case 2:
            y += "Charlie";
            y = "break";
        case 3:
            y += "Delta";
        default:
            y = "Echo" + y;
    }
    return y;
}
```

- (a) What is the value of `g("")` ?

SOLUTION: "Alpha"

- (b) What is a good test for `g("")` ?

SOLUTION: `assert(g("") == "Alpha");`

- (c) What is a good test for `g("3")` ?

SOLUTION: `assert(g("3") == "Bravo");`

- (d) What is a good test for `g("ab")` ?

SOLUTION: `assert(g("ab") == "EchobreakDelta");`

- (e) What is a good test for `g("san")` ?

SOLUTION: `assert(g("san") == "Alpha");`

- (f) What are valid argument values for x that would get g to return the string "EchoDelta"?

Describe the requirements of x in words.

SOLUTION: This is never possible for the current defined function g, the switch is subject to even-divisibility of the input string x by value 3 it is bound on 0 to 2 inclusive.

5. Reading Functions: guided:: The questions on this page ask about the recursive `foo` function defined below:

```
string foo( string x ) {  
    string result = "";  
    if ( x != "" ) {  
        istringstream y( x );  
        string z, remaining;  
        getline( y, z, ':' );  
        getline( y, remaining );  
        if ( z.length() % 2 == 0 ) {  
            result = z + ">" + foo( remaining );  
        } else {  
            result = foo( remaining );  
        }  
    }  
    return result;  
}
```

- (a) What is the base case of the `foo` function? What is a good test for this function call?

SOLUTION: `assert(foo("") == "");`

- (b) What is expected result of `foo("hi:there:sup:dude")`?

SOLUTION: `"hi>dude>"`

- (c) What is expected result of `foo(":1:23")`?

SOLUTION: `>23>"`

- (d) In your own words, describe the goal of the `foo` function. This is looking for you to define your own understanding and is not looking for an exact answer.

SOLUTION: Something about splitting a string by ':' delimiters, and keeping only the even length strings. Not looking for an exact answer, but making sure we have our own conceptual model of the function. **During exams, it is often a good idea to figure out what a function does at a high level before trying to work through the code by hand for a specific set of inputs.**

- (e) Write down notes about `getline` (shown above) vs `>>` (not shown).

SOLUTION: No solutions provided for this sample, it is up to you to verify your understanding of the topic using the resources that are available to you. Ask your peers, use office hours, use EdStem.

6. Reading helper functions and writing functions: self-study Use the provided `isOdd` function to solve a more interesting problem of your own choosing. (hint provided at top of next page)

```
bool isOdd( int y ) {  
    if ( y == 0 ) return false;  
    // ask yourself if the 'else' is necessary here  
    else if ( y == 1 || y == -1 ) return true;  
    if ( y > 0 ) return isOdd( y - 2 );  
    return isOdd( y + 2 );  
}
```

Ideas for approach:

- (a) Describe the task that could call `isOdd`
- (b) Define the task well and draw pictures of expected inputs and outputs
- (c) write the a function to solve your task. Be sure to call `isOdd` in your function.
- (d) now that you have completed this on paper, validate it in VSCode. Debug on paper as needed.

SOLUTION: No solutions provided for this sample, it is up to you to verify your understanding of the topic using the resources that are available to you. Ask your peers, use office hours, use [EdStem](#).

HINT FOR PREVIOUS PAGE: try appending a “.” to strings that are odd length, try counting the number of odd values in an array (available in solutions code), try removing all odd values from a vector, try it with a loop, try it with recursion, try it with a vector of structs that have ints rather than just a vector of int (available in solutions code).

7. writing functions: self-study Use this page to work through a function (or collection of functions) of your own design. If you can't think of a problem worth solving with a function, ask a peer, ask a member of the course staff (in office hours or on EdStem), ask an AI that you trust. Consider the same kind of “ideas for approach” that are described on the previous page.

SOLUTION: No solutions provided for this sample, it is up to you to verify your understanding of the topic using the resources that are available to you. Ask your peers, use office hours, use EdStem.

8. (static) arrays[]: guided:: The following questions rely on the provided code

```

void prependDont( string x[] [2], unsigned int len ) {
    for ( unsigned int i = 0 ; i < len ; i++ ) {
        x[i][1] = x[i][0];
        x[i][0] = "Don't";
    }
}
void printMat( string x[] [2], unsigned int len ) {
    for ( unsigned int row = 0 ; row < len ; row++ ) {
        for ( unsigned int col = 0 ; col < 2 ; col++ ) {
            if ( col != 0 ) { cout << " "; }
            cout << x[row][col];
        }
        cout << endl;
    }
}
int main() {
    string mat[4][2] = {
        { "stop", "overthinking" },
        { "think" }, { "procrastinate" }, { "give up" }
    };
    // YOUR CODE HERE
}

```

(a) What is the value of `mat[0][1][2]` for the `mat` defined in `main`?

SOLUTION: 'e'

(b) What does `prependDont` function do?

SOLUTION: Prepends the word "Don't" to each row of the matrix, limited to the `len` provided. Uses pass-by-reference to modify the parameter `x`.

(c) What does the `printMat` function do?

SOLUTION: prints the first `len` rows of the matrix for you. It happens to use pass-by-reference, but this particular function doesn't modify the matrix.

(d) What do you need to write in `// YOUR CODE HERE` to produce the following result using both of the functions provided? (yes, it's a silly thing to do, but it is something you have the skills for)

Don't stop
think

SOLUTION:

`prependDont(mat, 1);`
`printMat(mat, 2);`

Like we noted during the foo question, during exams, it is often a good idea to figure out what a function does at a high level before trying to

work through the code by hand for a specific set of inputs. On an exam, you might only see this question and not parts (b) and (c).

OBSERVE that pass by reference means prependDont will change mat for us

OBSERVE that using numbers smaller than 4 is valid and causes specific behavior.

9. (static) arrays[]: self study:: Construct your own problem and seek resources as needed. **Try to do something where the length matters and the use of pass-by-reference matters**

SOLUTION: No solutions provided for this sample, it is up to you to verify your understanding of the topic using the resources that are available to you. Ask your peers, use office hours, use EdStem.

10. Vectors: guided:: This page explores the problem of removing every even indexed element from vector of structs through a function named `rmEvenIndices`. An example is given below.

```
struct Hand {
    int value;
    string details;
};

vector<Hand> table;
table.push_back( { 11, "double down" } );
table.push_back( { 17, "troublesome hand" } );
table.push_back( { 21, "blackjack" } );
table = rmEvenIndices( table ); // index 0 and index 2 removed
// remaining: { 17, "troublesome hand" }
```

- (a) What are three good tests that you can write for the final `table` after `rmEvenIndices` is called shown above in a comment?

SOLUTION:

```
assert( table.size() == 1 );
assert( table.at( 0 ).details == "troublesome hand" );
assert( table.at( 0 ).value == 17 );
there are other options, but they are less obvious
```

- (b) Which of the following vector methods are likely helpful in solving `rmEvenIndices`: `at`, `begin`, `back`, `pop_back`, `push_back`, `insert`, and `erase`? Why? (not looking for exact answer, just getting out ideas.)

SOLUTION: “`pop_back`” and “`erase`” feel like good candidates. It might be possible with just “`at`”.

- (c) Implement `rmEvenIndices`. Start with the function signature, move to psuedo-code and finally write a working function. It can be recursive, it can use as many or as few of the methods described. Aim for something that is easy to read.

SOLUTION: there are MANY valid answers, below is just one of them. I write a few others in the published solution code

```
vector<Hand> rmEvenIndices1( vector<Hand> table ) {
    // using .at and .push_back
    vector<Hand> result;
    // note how this skips over the odds
    for ( unsigned int i = 1 ; i < table.size( ) ; i += 2 ) {
        result.push_back( table.at( i ) );
    }
    return result;
}
```

11. Vectors: self study:: Construct your own problem and seek resources as needed. **to optimize study, find a problem that uses the methods not used in the previous problem.**

SOLUTION: No solutions provided for this sample, it is up to you to verify your understanding of the topic using the resources that are available to you. Ask your peers, use office hours, use EdStem.

12. Streams: warm up

- (a) Describe the difference between an input stream and output stream.

SOLUTION: Input streams like `cin` are used to read data from the stream into the code. Output streams like `cout` are used to write data from the code to the stream.

- (b) Describe the difference between the following:

- `x >> y`
- `getline(x, y)`
- `getline(x, y, z)`

SOLUTION:

- In all of these cases `x` is an input stream that we are reading from. They all work on `cin`, `istringstream`, and `ifstream`.
- `getline` requires that `y` is a string, while `>>` allows `y` to be another type and will do type conversions of the string data for us if it can.
- `x >> y` reads until the first whitespace (space, tab, NEW_LINE), shoves that data into `y` without the whitespace attempting type conversions as needed. **It STOPS the read head BEFORE the white space (often want to pair with ignore)**
- `getline(x, y)` reads until the first NEW_LINE, shoves that data into `y` without the new-line as a string. **It STOPS the read head AFTER the new-line**
- `getline(x, y, z)` reads until the first `char z`, shoves that data into `y` without the `char z` as a string. **It STOPS the read head AFTER the char z**

- (c) Let there be an input stream named `x` with the following data:

```
hello    my name is
Gregor
and more stuff is here
```

We run the following code:

```
string y0, y1, y2;
x >> y0; // y0 is "hello"
// CHANGE CODE
getline( x, y1, 'g' ); // y1 is "    my name is<NEW_LINE>Gre
getline( x, y2 ); // y2 is "or"
```

State in your own words why each of the following are true

- `y0` does not have a space at the end.
- `y1` has a three spaces at the beginning. (followup: what could you write in “CHANGE CODE” to remove that whitespace?)

- `y1` includes a newline in the middle of it
- `y1` doesn't end with the letter 'g'.
- `y2` doesn't begin with the letter 'g'.
- `y2` doesn't end with a newline.

SOLUTION: See explanations in previous question to answer these questions. Explore it in your preferred coding environment to better understand this topic.

- (d) What is different about opening an `ifstream` vs an `ofstream`?

SOLUTION: If you open an `ofstream` and the file does not already exist, the library will make the file for you as long as it has permission to do so in that location on your computer. This is not true for `ifstream`. Note, you still need to do a check that the file is open due to system permissions and other uncommon issues.

- (e) Why is it important to close a file stream?

SOLUTION: Failure to close any file stream, can result in a security vulnerability that allows bad actors to mess with your data. Specific to output, failure to close `ofstream` may lead to the data you wrote not actually being saved to the file, so your code took time to run, but didn't really get the job done.

13. Streams: self study:: Construct your own problem and seek resources as needed.**NOTE:** streams covered this semester include `iostream`, `fstream`, and `sstream`. Note the provided array problem tests `getline`, consider writing a problem that works with input and output streams and uses `>>` instead of `getline`. (hint provided at top of next page) **SOLUTION:** No solutions provided for this sample, it is up to you to verify your understanding of the topic using the resources that are available to you. Ask your peers, use office hours, use EdStem.

HINT FOR PREVIOUS PAGE Write a function, given two file names as strings, read data from the first file, write the data to the second file, but remove all lines which begin with the phrase “COMMENT :”. Do not change the first file. Use best practices to open and close files.

SOLUTION: code available in solution code document... be sure to write a valid in.txt file and open the out.txt file to validate if the code actually works.

14. Structs/Instances: warm up

- (a) Following best-practice in C++, should a `struct` contain member functions?

SOLUTION: No. Sure, you can technically do it, but it's strongly NOT recommended, use a `class` instead.

- (b) Following best-practice in C++, should a `struct` contain “private” data?

SOLUTION: No. Sure, you can technically do it, but structs have default access modifier “public” and should really only have public non-function data. If you want private data, use a `class` instead.

- (c) When passing a struct to a function, does it default to pass-by-value or pass-by-reference?

SOLUTION: pass-by-value. Only arrays are pass-by-reference.

- (d) When passing an array of structs to a function, does it default to pass-by-value or pass-by-reference? Why?

SOLUTION: pass-by-reference. arrays are pass-by-reference. It doesn't matter what the type of the elements of the array are, it's still an array.

- (e) When passing a struct to a function that contains an array as a data member, does it default to pass-by-value or pass-by-reference? Why?

SOLUTION: pass-by-value. The full struct is passed, a struct is not an array even if it holds an array inside it.

- (f) Consider a struct that has an array as a data member accessed `instance.array`, when calling a function with argument `instance.array`, does it default to pass-by-value or pass-by-reference? Why?

SOLUTION: pass-by-reference. We are no longer passing the struct to the function, we are passing the internal array to the function.

- (g) When passing an array of structs to a function that contains an array as a data member, does it default to pass-by-value or pass-by-reference? Why?

SOLUTION: pass-by-reference. Note a few important points here:

1. The internals of the struct having their own arrays does not matter. That is a mislead in the question. **Learning what is and is not important when reading a question can save you a lot of time during an exam. Ask on EdStem for more examples from your past exams.**
2. The type of each element of the outer array does not matter.
3. We are passing an array to a function. Arrays are pass-by-reference.

15. Structs/Instances: self study:: Construct your own problem and seek resources as needed.

SOLUTION: No solutions provided for this sample, it is up to you to verify your understanding of the topic using the resources that are available to you. Ask your peers, use office hours, use EdStem.

16. Classes/Objects: guided:: let us play a game with friends. Before reading further decide on who will be the “game lead(s)” and “adventurer(s)” (need at least one of each). Now, only the game lead(s) should read further....

1. Opening dialog read by the game lead(s): “In the heart of the arcane sanctum, a series of ancient number glyphs lie etched into stone boxes, each pulsing faintly with enchanted light. A mysterious spell binds them—an enchantment I know well, yet am bound by oath and magic to speak nothing of its nature. Your quest is thus: brave the puzzle, unravel the pattern, and uncover the hidden magic woven within. Only through trial and cunning can you pierce the veil and reveal the secret of the spell. Will you accept this challenge, adventurer?” (credit ChatGPT)
2. Explain to the adventurer(s) their available moves:
 - (a) “explore a new glyph”
 - (b) “view the glyph”
 - (c) “apply pressure to the the glyph <quantify with a number>”
 - (d) “reduce pressure to the the glyph <quantify with a number>”
 - (e) attempt to describe the magic spell
3. Don’t share the following information with the adventurer(s). The game lead(s) available moves are as follows:
 - (a) if the adventurer wishes to explore a new glyph, ignore all known number boxes, call the default constructor to make a new object to observe.
 - (b) if the adventurer wishes to view the glyph, you must tell them the current number.
 - (c) if the adventurer applies pressure to the glyph, you must call the setter with a positive number and apply the rules of the `_magic` function. Tell the player when you are ready to continue, but do not tell them the value of the glyph or show them any of your math.
 - (d) if the adventurer reduces pressure to the glyph, you must call the setter with a negative number...
 - (e) if the adventurer describes the magic spell incorrectly, you must tell them to keep playing the game. **Add any lore you wish.**
 - (f) if the adventurer describes the magic spell correctly, make up your own ending for the game.
4. all code is on the next page for the game lead(s) to use to facilitate the game.
5. after the game ends, review the code together. Review topics like **default vs parameterized constructors**, **data members (attributes) vs member functions (methods)**, **public vs private (access modifiers)**, **getters vs setters**, **class vs object**, the differences in application and best practices between **struct vs class**. Write a **parameterized constructor** for the NumBox class that sets the value and calls the private `_magic` function.

SOLUTION: the code is available for download. You can write a small driver to play against the computer. You can take turns writing new `_magic` functions to see if the other player can guess your magic spell(s).

17. Classes/Objects: self study:: Construct your own problem and seek resources as needed.

NOTE: vectors and their member functions are great example of a well defined class. The vector class defines a series of member functions for you to use to access complex memory without having to worry about how the memory is handled

SOLUTION: No solutions provided for this sample, it is up to you to verify your understanding of the topic using the resources that are available to you. Ask your peers, use office hours, use EdStem.

18. nesting concepts: warmup

- (a) What data types default pass-by-reference?

SOLUTION: arrays

- (b) What data types default pass-by-value?

SOLUTION: all other data types

- (c) What data types use dot-notation? Give an example.

SOLUTION: struct instances derived from structs (sometimes called objects) and objects derived from classes. RECALL best-practice is that all struct members are public while classes make encapsulate data members (attributes) as private and instead access and manipulate the data through public member functions (methods) called getters or setters. This includes imported classes like “vector”, “string stream”, and “file stream”.

e.g.

```
vector<int> v ; v.push_back( 5 );
```

- (d) What data types use bracket-notation? Give an example.

SOLUTION: arrays and strings

e.g.

```
string arr[5] = "hello", "there" ; arr[0][4];
```

- (e) Following C++ best practice, how do you access data from a struct instance? Give an example.

SOLUTION: Use dot notation. Access the data directly.

e.g. let MyStruct be a defined struct with int member “x”

```
MyStruct instance = 4 ; cout << instance.x << endl;
```

- (f) Following C++ best practice, how do you access data from a object? Give an example.

SOLUTION: Use dot notation. Access the encapsulated private data member (attribute) through a public member function (attribute)[getter].

e.g. let MyClass be a defined class with private int data member “_y” and public member function “getY” and a paramterized constructor

```
MyClass object(4); cout << object.getY( ) << endl;
```

19. Nesting concepts: self study:: Construct your own problem and seek resources as needed.

NOTE: any of the self-studies earlier in this packet are great candidates to practice nesting. A pinnacle problem to solve may involve a class that encapsulates a vector of 2-D arrays of structs that include strings resulting in accesses requirements like:

```
o.getVector().at( 0 )[ 1 ][ 2 ].stringMember[ 3 ];
```

You could also consider having streams instead of strings at the final layer like:

```
getline( o.getVector().at( 0 )[ 1 ][ 2 ].streamMember, someString, ':' );
```

Then consider different orderings of this nesting and various functions to deal with pass-by-reference vs pass-by-value behavior. Throw in a class for good measure.

RECALL: The questions in this study guide are designed to be difficult and help you identify knowledge gaps that warrant further study.

SOLUTION: No solutions provided for this sample, it is up to you to verify your understanding of the topic using the resources that are available to you. Ask your peers, use office hours, use EdStem.