Unit 5 Study Guide and Review: Memory Management and Testing

5a. Define test cases and coverage analysis

1. What are the types of unit tests and how they are used?

White-box Tests

These tests are designed by examining the internal logic of each module and defining the input data sets that force the execution of different paths through the logic. Each input data set is a test case.

Black-box Tests

These tests are designed by examining the specification of each module and defining input data sets that will result in different behavior (e.g. outputs). Black-box tests should be designed to exercise the software for its whole range of inputs. Each input data set is a test case.

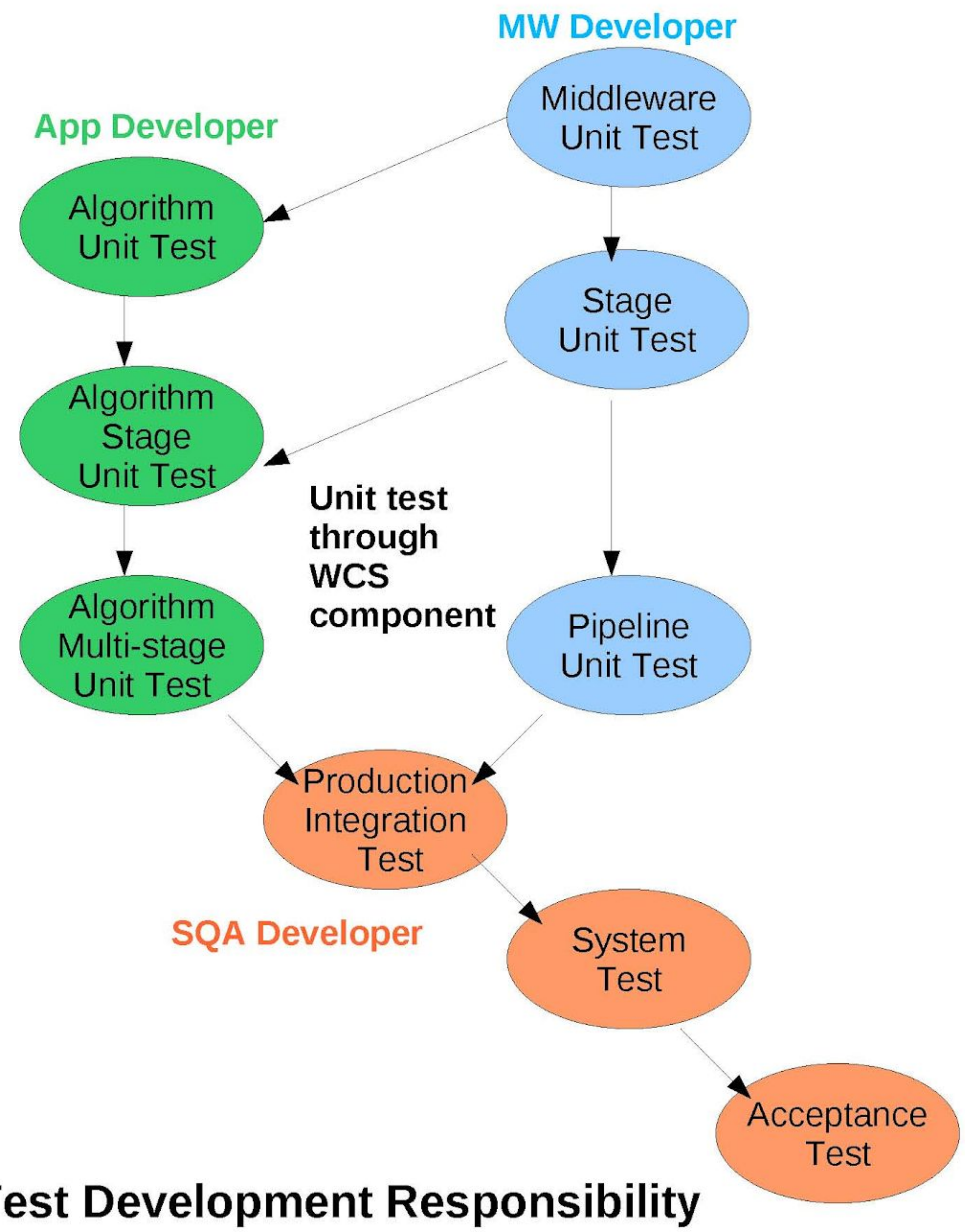
Performance Tests

If the detailed design placed resource constraints on the performance of a module, compliance with these constraints should be tested. Each input data set is a test case.

The collection of a module's white-box, black-box, and performance tests is known as a Unit Test suite. A rough measure of a Unit Test suite's quality is the percentage of the module's code which is exercised when the test suite is executed.

2. What are the different roles in test development?

In essence, the Applications group, which is responsible for all science algorithm development, also develops the unit testers for: algorithm components, a stage wrapped algorithm, and the sequences of stage wrapped algorithms comprising a simple-stage-tester algorithm pipeline (i.e. without parallel processing job management). The Middleware group, so named because it is responsible for all low level framework software, develops the unit testers for those framework modules. Finally, the SQA team is responsible for the higher level testers for integration, system performance, and acceptance testing.



Test Development Responsibility

You should make sure that each and every branch of decisions are reached during your test cases to ensure that all possible outcomes are tested. The goal is to try and see if you can get your program to crash, so that if it does, you can fix it. Test for extreme cases, user error, bad input and any other possibility. When developing test cases, many people play a role in this process, as it is essential that there be an understanding of how users might use the software. For more information on unit tests, refer to this page.

5b. Use debugging tools

1. What are the steps to the debugging process?
    1) Identify is the first step in the debugging, it means discover what is the bug and why it happen
    2) Isolate is a step to separate the buggy code with other healthy code.
    3) Fix is a step to correct the buggy code; This step is the core part of debugging.
    4) Review is the final step and is easily forgotten by us; Review is a step to ensure the bug you fixed are working correctly now. Also the changes are not affecting other parts of the application and working properly in the target platform.
2. What are the shortcut keys to step through your coding?

Key	**Description**
`F5`	Executes the currently selected line and goes to the next line in your program. If the selected line is a method call the debugger steps into the associated code.
`F6`	F6 steps over the call, i.e. it executes a method without stepping into it in the debugger.
`F7`	F7 steps out to the caller of the currently executed method. This finishes the execution of the current method and returns to the caller of this method.

`F8`	F8 tells the Eclipse debugger to resume the execution of the program code until is reaches the next breakpoint or watchpoint.

Logical errors are often very difficult to figure out. Using the debugging tools that are build into your IDE, in this case, Eclipse, allows you to track your data values and see what is happening behind the scenes. Think of it to a backstage pass to the code. Running the code in debug mode allows you to review each line of code, and the resulting data values. For more information on debugging tools, refer to this tutorial.

5c. Use memory management

1. How do you write a method that deallocates memory from a class?

   A **destructor** is called for a class object when that object passes out of scope or is explicitly deleted. A **destructor** is a member function with the same name as its class prefixed by a ~ (tilde)

2. How do you release a pointer from memory?
   a. Delete is the C++ statement used to deallocate dynamic memory
      i. New is the C++ keyword used to allocate dynamic memory

Whenever you instantiate a class, you are reserving memory space, once you are no longer using that class, it is important that you remove the instance from memory, freeing up memory space for use later. The same practice should be done with the use of pointers, removing the pointer allocation from memory. For more information on memory management, refer to these slides.

Dynamic memory

In the programs seen in previous chapters, all memory needs were determined before program execution by defining the variables needed. But there may be cases where the memory needs of a program can only be determined during runtime. For example, when the memory needed depends on user input. On these cases, programs need to dynamically allocate memory, for which the C++ language integrates the operators new and delete.

Operators new and new[]

Dynamic memory is allocated using operator new. new is followed by a data type specifier and, if a sequence of more than one element is required, the number of these within brackets []. It returns a pointer to the beginning of the new block of memory allocated. Its syntax is:

pointer = new type
pointer = new type [number_of_elements]

Operators delete and delete[]

In most cases, memory allocated dynamically is only needed during specific periods of time within a program; once it is no longer needed, it can be freed so that the memory becomes available again for other requests of dynamic memory. This is the purpose of operator delete, whose syntax is:

```
delete pointer;
delete[] pointer;
```

The first statement releases the memory of a single element allocated using new, and the second one releases the memory allocated for arrays of elements using new and a size in brackets ([]).

The value passed as argument to delete shall be either a pointer to a memory block previously allocated with new, or a *null pointer* (in the case of a *null pointer*, delete produces no effect).

Unit 5 Vocabulary
This vocabulary list includes terms that might help you with the review items above and some terms you should be familiar with to be successful in completing the final exam for the course.
Try to think of the reason why each term is included.

- Breakpoint
  - A breakpoint is set on an executable line of a program. If the breakpoint is enabled when you debug, the execution suspends before that line of code executes.
- Coverage analysis

The purpose of coverage analysis is to verify the thoroughness of a test suite. For example, unit testsare used to validate the implementation of detailed design objects through comprehensive testing. Coverage analysis checks that the testing is, indeed, comprehensive by executing instrumented unit tests which records the complete execution path through the code and then calculating metrics indicative of the coverage achieved during execution.

Coverage analysis examines the output of a code instrumented to record every line executed, every conditional branch taken, and every block executed. It then generates metrics on:

- Percent of statements executed
- Percent of methods (and/or functions) executed
- Percent of conditional branches executed
- Percent of a method’s (and/or function’s) entry/exit branches taken.

The metrics give a general idea of the thoroughness of the unit tests.

- Dangling pointer
  - **Dangling Pointer**. If any **pointer** is pointing the memory address of any variable but after some variable has deleted from that memory location while **pointer** is still pointing such memory location. Such **pointer** is known as **dangling pointer** and this problem is known as **dangling pointer**problem.
- Debug mode
  - A debug menu or debug mode is a user interface implemented in a computer program that allows the user to view and/or manipulate the program's internal state for the purpose of debugging.
- Debug perspective
  - allows you to manage the debugging or running of a program in the workbench. It displays the stack frame for the suspended threads for each target you are debugging. Each thread in your program appears as a node in the tree. It displays the process for each target you are running.
  - If the thread is suspended, its stack frames are shown as child elements.

    destructor
- Debugging
  - the process of identifying and removing errors from computer hardware or software.
- Debugging tools

    - standard **debugging**functionality, including the ability to perform step execution, to set breakpoints and values, to inspect variables and values, and to suspend and resume threads.
- Destructor
    - A **destructor** is called for a class object when that object passes out of scope or is explicitly deleted. A **destructor** is a member function with the same name as its class prefixed by a ~ (tilde)
- Detail Formatter
    - Eclipse has a feature called "Detail Formatter" where for each class that you are debugging, you can specify how it should display.
- Heap
    - The **heap memory** segment is an area of **memory** used for dynamic allocations, **meaning** that blocks of **memory** can be allocated and freed in an arbitrary order and accessed multiple times (as opposed to the stack, which is Last-In-First-Out).
- Memory leak
    - a memory leak is a type of resource leak that occurs when a computer program incorrectly manages memory allocations in such a way that memory which is no longer needed is not released.
- Memory management

A good understanding of how dynamic memory really works in C++ is essential to becoming a good C++ programmer. Memory in your C++ program is divided into two parts −

- The stack − All variables declared inside the function will take up memory from the stack.
- The heap − This is unused memory of the program and can be used to allocate the memory dynamically when program runs.

Many times, you are not aware in advance how much memory you will need to store particular information in a defined variable and the size of required memory can be determined at run time.

You can allocate memory at run time within the heap for the variable of a given type using a special operator in C++ which returns the address of the space allocated. This operator is called new operator.

If you are not in need of dynamically allocated memory anymore, you can use deleteoperator, which de-allocates memory that was previously allocated by new operator.

  -

- Stack
  - **C++** the **stack memory** is where local variables get stored/constructed. The **stack** is also used to hold parameters passed to functions.
- Stop point
  - A place to pause the debugger; Breakpoints and watchpoints are types of stop points
- Test case
  -
- Toggle breakpoint
  - Turn the breakpoint off or on
- Unit test
  - a level of software **testing** where individual **units**/ components of a software are tested. ... A **unit** is the smallest testable part of any software. It usually has one or a few inputs and usually a single output. In procedural programming, a **unit** may be an individual program, function, procedure, etc.
- Variables view
  - The **Variables View** displays information about the variables associated with the stack frame selected in the **Debug View**.
- Watchpoint
  - A watchpoint is a special breakpoint that stops the execution of an application whenever the value of a given expression changes, without specifying where it might occur. Unlike breakpoints (which are line-specific), watchpoints are associated with files. They take effect whenever a specified condition is true, regardless of when or where it occurred. You can set a watchpoint on a global variable by highlighting the variable in the editor, or by selecting it in the Outline view.