**Debugging**

* Debugging is the process of finding and fixing errors in a program (usually logic errors, but can also include compiler/syntax errors)
* For syntax errors, understand what the compiler is telling you .
* Always focus on fi xing the first problem detected.
* Can range in complexity from fixing simple errors to collecting large amounts of data for analysis.
* The ability to debug by a programmer is an essential skill (problem solving) that can save you tremendous amounts of time (and money)
* Maintenance phase is the most expensive phase of the software life cycle.
* Understanding that bugs are unavoidable.

**Common Problems**

* Logic Errors
* Syntax Errors
* Memory Corruption
* Performance / Scalability
* Lack of Cohesion ( too much functionality)
* Tight Coupling (Dependencies)

**Objective in Code is very high Cohesion and low Coupling .**

**Debugging Process**

* Understanding the problem(Sit down tester, understand requirements)
* Reproduce the problem
* Sometimes very difficult as problems can be intermittent or only happen in very rare circumstances.
* Parallel processes or threading problems.
* Simply the problem/ Divide and conquer / isolate the source
* Remove parts of the original test case.
* Comment out code / back out changes.
* Turn a large program into a lot of small programs (unit testing).
* Identify origin of the problem (in the code)
* Use Debugging Tools if necessary.
* Solve the problem
* Experience and practice
* Sometimes includes redesign or refactor of code.

**Test ! Test ! Test ! Test ! Test ! Test !**

**Techniques and Tools**

* Tracing / Using print statements
* Output values of variables at certain points of a program.
* Show the flow of execution.
* Can help isolate error
* Debuggers - monitor the execution of a program, stop it , restart it , set breakpoints and watch variables in memory.
* Log Files - can be used for analysis, add “good” log statements to your code.
* Monitoring Software - run - time analysis of memory usage, network traffic, thread and object information.

**Common Debugging Tools**

* Exception Handling helps a great deal to identify catastrophic errors.
* Static Analyzers - analyze source code for specific set of known problems.
* Semantic checker, does not analyze syntax.
* Can detect things like uninitialized variables, memory leaks , unreachable code, deadlocks or race conditions.
* Test Suites - run a set of comprehensive system end-to-end tests
* Debugging the program after it has crashed .
* Analyze the call stack.
* Analyze memory dump (core file ).

**Preventing Errors**

* Write high quality code (follow good design principles and good programming practices )
* Unit tests - automatically execute when compiling
* Helps avoid regression.
* Finds errors in new code before it is delivered.
* TDD (test Driven Development)
* Provide good documentation and proper planning (write down design on paper and utilize pseudo code)
* Work in Steps and constantly test after each step.
* Avoid too many changes at once.
* When making changes , apply them incrementally. Add one change , then test thoroughly before starting the next step.
* Helps reduce the possible sources of bugs, limits problem set.

**Understanding the call stack**

* A stack trace (call stack) is generated whenever your app crashes because of a fatal error. Stack = Last data added in, is the First data out LIFO...
* A stack trace shows a list of the function calls that lead to the error.
* Includes the filenames and line numbers of the code that cause the exception or error to occur.
* Top of the stack contains the last call that caused the error (nested calls )
* Bottom of the stack contains the first call that started the chain of calls to cause the error
* You need to find the call in your application that is causing the crash.
* A Programmer can also dump the stack trace.

**Common C Mistakes**

* Misplacing a semicolon.

If (j ==100);

J = 0;

* The value of j will always be set to 0 due to the misplaced semicolon after the closing parenthesis.
* Semicon is syntactically valid (it represents the null statement), and , therefore , no error is produced