Debugging - overview

- 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 fixing 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)
- 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.
- Understand that bugs are unavoidable.

Common Problems

- Logic Errors
- Syntax Errors
- Memory Corruption
- Performance / Scalability
- Lack of Cohesion
- Tight coupling (Dependencies)

Debugging Process

- Understand the problem (sit down with 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

- Simplify 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 smaller 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

Techniques and Tools

- Tracing / using print statements
    - ➔ Output values of variables at certain point of a program
    - ➔ Show the flow of execution

    - ➔ Can help isolate the error
- Debuggers - monitors 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 Analyzer - analyze source code for specific set of known problems
    - ➔ Semantic checker , does not analyze syntax
    - ➔ Can dect 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 and good programming practices)
- Unit tests - automatically execute when compiling
    - ➔ Helps avoid regression
    - ➔ Finds errors in new codes before it is delivered
    - ➔ TDD (Test driven Development)
- Provide good documentation and proper planning (write down design on paper and utilize pseudocode)
- 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 source of bugs, limited problem set.