Kas's Software Engineering Practice Test of Doom

Please note any answers in () Typically are not part of the answer rather are added context or a joke.

1 Intro to SWE

1. What is engineering? The creative application of scientific prinicples to design or develop structures.

2. What is software Engineering? The application of a systematic, disciplined, quantifiable approach to the developement, operation and maintenance of software, and the study of these approaches. (or without the yap: the application of engineering to software).

3. What factors affect SW quality.
   Complexity and change.

4. How does complexity affect quality?.
   When a system becomes so complex a singular programmer can no longer understand it.

5. How does change affect quality?
   The entropy of a software system increases with each change. Each implemented change erodes at the structure of the system. Making the next change more expensive. As time goes on the cost of implementing changes will increase.

6. What are the methods of dealing with complexity? Abstraction, Decomposition, and hierarchy.

7. What are the ways of abstracting through modeling?
   System mode, Task model, and Issue model.

8. Give an exmaple of a System model and what it does.
   Object model: What is the struct of the system.
   Functional Model: What are the functions of the system and how does data flow through the system.
   Dynamic Model: How does the system react to external events.

9. give an example of a task model and what it does.
   Gantt/PERT chart: What are the dependecies between each task.
   Schedule: How can this be done within a time limit.
   Org Chart: What are the roles in the project.

10. What is an issues model?
    What are the open and closed issues. What constraints were posed by the client? What resolutions were made?

11. What is decomposition?
    A technique used to master complexity (divide and conquer type stuff).

12. What are each of the types of decomposition?
    Functional decomposition and Object-oriented decomposition.

13. What is hierarchy.
    Providing simple relationships between different chunks.

14. what are the two important hierarchies?
    Part of and is kind of.

2 Lifecycles

1. What is a Software developement Life cycle? A set of activites and their realtionships to ea chother to support the developement of software.

2. What was the first life cycle?
   Waterfall model (wooo)

3. What are the benefits of the waterfall model?
   Nice milestones, No need to look back, one activity at a time, and easy to check progress.

4. what are thecons of a waterfall model?
   Software developement tends to not be linear and you'll need to go back and refine.

5. When should one use a prototyping model?
   When user requirements are unclear. For systems with high emphasis on the UI and require a high degree of user interaction.

6. What are the pros of a prototype model? Iterative process, frequent feedback, fast.

7. What are the cons of a prototype model?
   Design compromise frequently due to time constraints, customer could think it's a real product. Lack of robustness and completeness and at the end of each iteration we have a prototype.

8. what are the two types of prototype and describe them.
   Functional prototype: implement and deliever an operational system with minimum funcftionally and build from there.
   Exploratory prototype: Implement part oft he system to learn more about the requirements.
   Revolutionary prototyping: Get user expierence with a throwaway version to get the requirements right then build the whole system
   Evolutionary Prototyping: Used as the basis for the implementation of the final system.

9. What is the disadvantage of evolutionary prototyping?
   Can only be used if target system can be constructed in a prototyping language.

10. What is the spiral model? Use a waterfall model for each prototype developement (a cycle). If risk has been resolved evaluate the results of the cycle and plan for the next. If you run into an unsolvable product terminate (give up all roads leave to doom).

11. what are the advantages of a spiral model?
    Model runs through the entire lifecycle of software and is iterative.

12. what are the disadvantages?
    Difficult to convince customers the process is easily manageable. Project length cannot be determined.

13. When is a SW developement process considred mature?
    If the develpement activites are well defined and if management has some control over the quality, budget, and Schedule in the project.

14. what are the 5 maturity levels?
    Initial Level (ad hoc/chaotic)
    Repeatable level the process depends on individuals called champions.
    Defined level the process is institutionalized (sanctioned by management).
    MAnaged level activties are measured and provide feeback for resource allocation however the process is not changing.
    Optimizing level process allows feedback of information to change the process itself.

15. What is agile based on.
    It is difficult to predict in advance which requiprements or customer priorities will change.
    For many types of software design and construction activities are interleaved.
    Analysis, design, and testing are not as predictable from a planning perspective.

16. Whatare the principles of agile?
    Highest priority is to satisfy the customer with early and continous deliever of valuable software.
    Welcome changing requirements anywhere in development.
    Delievering software frequently with a preference for shorter delievery schedules
    Business people and developers must work togheter daily
    Face-to-face communcation is the most effective method of conveying informaiotn within the developement team.

17. What is DevOps?
    The union of people, process, and products to enable continous delievery of value to end users.

18. what is the gaol of dev ops?
    To shorten developement time.

3 Modeling with UML

1. What does modeling help achieve?
   building an abstraction of reality. Ignores irrelatant detials through abstractions.

2. Why model software?
   Software is becoming increasingly more complex. Code is not easily understandable by developers who did not write it. We need simplier representations for complex systems.

3. what is a Phenomenon?
   An ojbect in the world of a domain as you percieve it.

4. what is a concept?
   Describes the properties of phenomena that are common.

5. What is an application domain?
   The enviroment in which the system is operating.

6. what is a solution domain.
   Technologies avaible to build the system.

7. What is a class diagram?
   Describe the static structure of a system.

8. What is a use case diagram.
   Describe the functional behavior of the system as seen by the user.

9. what is a sequence diagram.
   Describe the dynamic beahvior between actors and the system between objects of a system.

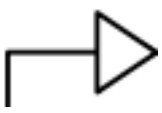
10. what is a state chart diagram.
    Describe the dyanmic behavior of an individual object (DFA!).

11. What is an activity diagram?
    Mdeol the dynamic behavior of a system, in particular the workflow.

12. what is the extends relationship in Use cases and how is it represented?
    It is represented by ¡¡extends¿¿ over the arrow pointing towards what it's extending. Extending represent exception cases in a use case.

13. what is the includes relation in Use cases and how is it represented?
    Similar to extends but ¡¡includes¿¿. Represents behavior that is factored out of the use case. beavhior is factored out for reuse.

14. what is the inherit relationship and how is it represented?



Relationship used to specialize another more generalized use case.

15. what does a class represent?
A concept.

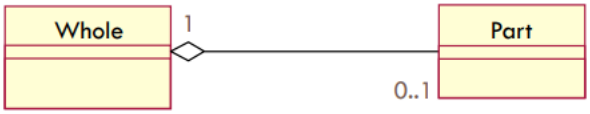
16. what are the parts of a class?
Attributes, Operations, Name, and signature.

17. how do you represent a One or more assocation?
1..*

18. How would you represent a specfied range?
2..7

19. How would you represent disjoint ranges?
2, 3..8

20. what is aggregation and how is it drawn?



A special case of assocation denoting a consists of hierarchy. The aggregate is the parent class and the componets are the child classes.

21. What does a solid diamond represent?
Composition, a stronger form of aggregation where componets cannot exist without the aggregate(parent).

22. What is inheritance in Class models?
Children classes inherit the attrbutes and operations of their parent class. Helps by eliminating redundancy.

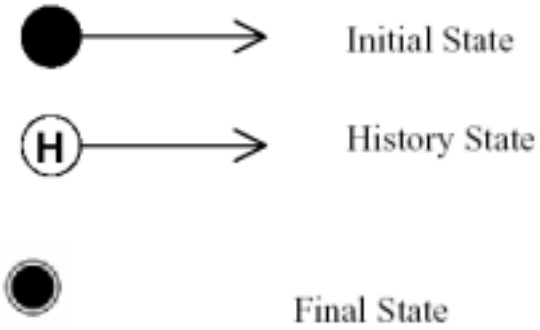
23. In UML what is a package?
A UML mechanism for organizing elements into groups. They are the basic grouping construct you can organize UML models with.

24. How is interation deonted in Sequence diagrams?
with a *.

25. How are conditions denoted in sequence daigrams?
With a boolean expression in [] before the messages name.

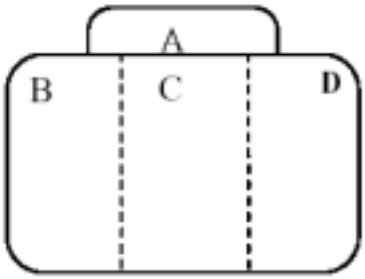
26. How is creation and destruction represented in UML?
Denoted with a message pointing towards the object at the start for creation. Destruction is denoted by a message with an X.

27. How do you draw an initial state, history state, final state?



28. How do you represent a state in a state chart?
With a rounded box with the states name in the top.

29. How do you represent an Orthgonal state?



30. T/F for a state to be active the state of the parent doesn't matter (Hamdy quickfire round).
False, the parent must be active.

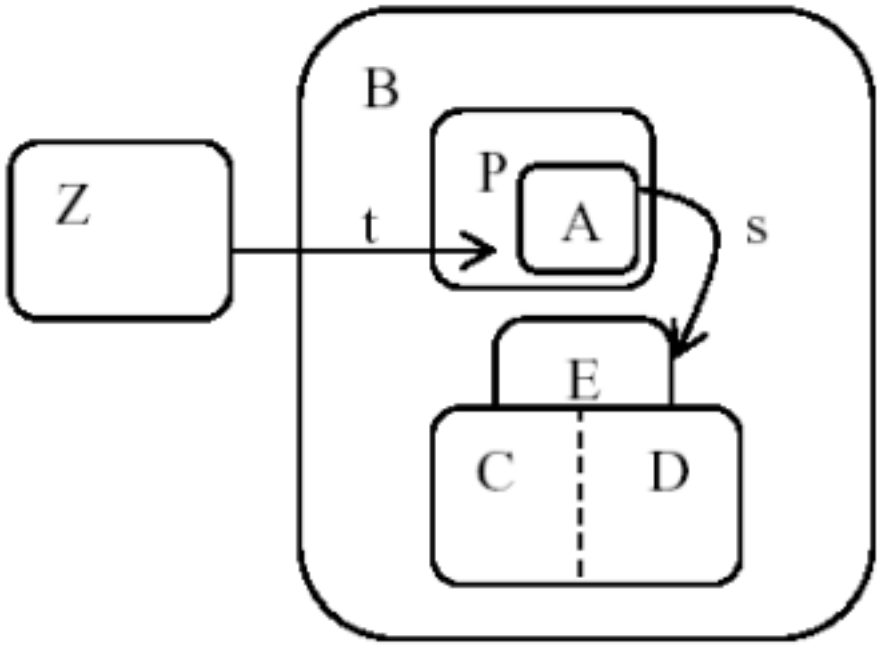
31. T/F at most one state of a non-orthogonal state can be active at a time.
True.

32. T/F Only one orthogonal state can be active at a time.
False all are active.

33. T/F threads on orthogonal states are orthogonal states as well.
False, each thread is a non-Orthgonal state.

34. T/F when an event occurs the source must be active for the event labeled transition to occur.
True.

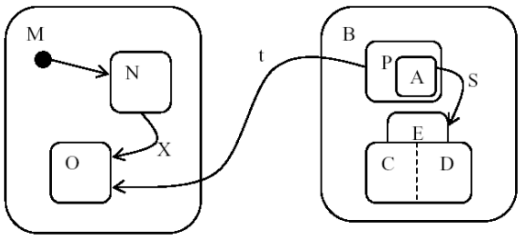
35. T/F when activating a destination state the ancestors need be activated.
False, they must be activated.



When t occurs
both P and B sh

36. T/F Each non-Orthogonal state may contain a history state, initial and/or final state.
True.

37. T/F When activating a non-orthononal state we must also activate the child state indicated by the initial state unless the containing non-orthogonal state is the ancestor of our destination state.
True.

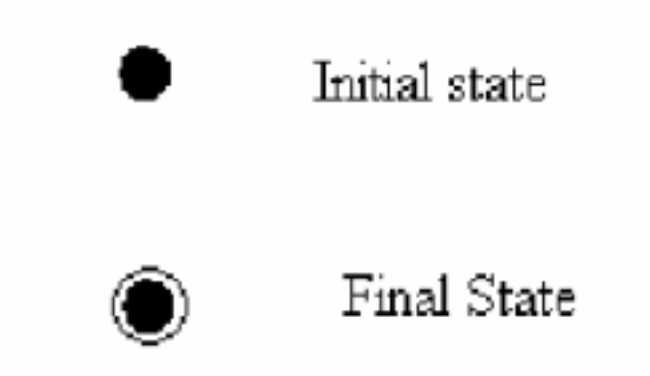


B, P and A are active and t happens

a. Deactivate B, P and A
b. Activate M and O, ignoring N according to above rule.

38. How does a history state work?
Remembers the child state last active within the containing non-orthogonal state.

39. In an activity daigram how do you draw an initial state and final state?

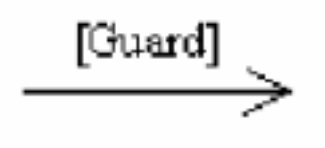


40. How do you represent a decision/merge elements in an activity diagram?
    With a non-solid diamond.

41. How do you fork/Join in an activity diagram?
    With a solid black bar and the branching elements.

42. how do you identify an action in an activity diagram?
    With a rounded rectangle box with the action inside.

43. How do you draw a transition?



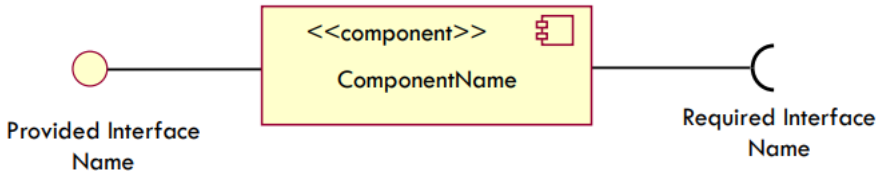
44. What is the guard in a transition?
    A boolean expression.

45. How do you draw an accept signal and send signal?

46. What is a component diagram?
Shows the organizations and dependecies among componets. Component diagram hides its implementation. Specifies the physical dependcy to interfaces it requires.

47. What is the difference between a component diagram and class diagram?
Components are logical abstractions while classes are physical things.

48. draw a sample component diagram.



49. What is a deployment diagram?
A diagram that shows configuration of processing nodes at run time. Communcation between these nodes and deployed artifacts that reside on them.

50. What does a connector represent?
A communcation method be it physical or a software protocal (or psychic).

4 Requirement engineering

1. what shape represents a component?
A 3-d Cube.

2. What makes a good problem statement?
The current situation, the funcftionally of a new system, the enviroment the system will be deployed in, delieverables expected by the client, delievery dates, and a set of Acceptance criteria.

3. What does FURPS stand for?
Functionality, Usability, Reliability, Perforamnce, and Supportability.

4. What are the parts of Functionality in FURPS?
Capabilities, Reusability, security.

5. What are the parts of Usability in FURPS?
Human factors, aesthetics, consistency, documentation.

6. What are the parts of Reliability?
Frequency/severity of failures, recoverability, predictability, accuracy.

7. What are the parts of perforamnce in FURPS?
Speed, resource usage, throughput, response time.

8. What are the parts of support in FURPS?
   Testability, extensibility, adaptability, maintainability, compatiability, configurability, servicability, installiability, robustness.

9. What is requirement engineering?
   Process of identifying, eliciting, analyzing, specifincg, validation, and managing the needs an expectations of the client.

10. What are the 3 activities of requirement engineering?
    Elicitation: definition of the systems understood by the customer.
    Analysis: Technical specification of the system in terms understood by the developer.
    Validation: Validation and verifcation.

11. What are the approaches to determine classes in objects.
    Application domain approach Ask application domain expert to identify relevant abstractions.
    Syntatic approach starting with use cases and extract objects from flow of events. Can also use noun-verb analysis to identify object model.
    Design patterns approach using reusable design patterns.
    Component based approach identify existing solution classes.

12. What are the 3 object types.
    Entity objects represent the persistent information tracked by the system.
    Boundary objects represent the interaction between the user and the system.
    Control objects represent the control of tasks performed by the system.

13. How do you implement dynamic modeling?
    Start with a use case or scenario. Model interaction between obejects (sequence diagrams!!!).
    Model dyanimc behavior of a single object (statechart diagram!!!).

5 Requirement Validation

1. What is requirement validation?
   The process onf confirming that the documeneted requir4ements match the customers needs.

2. What are 3 criteria of RE?
   Correctness: Is a true statement something the system must do.
   Completeness: Describes all significant requirements of concern to the user.
   Consistentcy: Does not conflict with other requirements.
   Unambigousness: Is subject to one and only one interpretation.
   Verifiability: Can be tested cost effectively.
   Modifiability: Changes do not affect the structre and style of the set.
   Traceability: The origin of each requirement can be found.
   Understandability: Comprehended by users and developers.

3. How do you determine if a requirement is veriabile?
   There exists a finite, cost effective proccess which the system can be check that requirements are meet.

4. Whuat are 2 ways of Verifiation and validation?
   Simple checks, prototyping, functional test design, reviews and inspections, and model based verifcation and validation?

5. What is the 1-10-100 rule?
   It is a rule that the longer a change is made within the product the more said change will cost (Minecraft players hate this rule).

6. What is a project agreement?
   Represents an acceptance of the analysis model by the client.
   Allows the client and developers and the client to converge on a single idea that agree about functions and features that the system will have.

7. When most priorities be decided?
   All of them must be addressed during analysis however, Medium and high priorities must be addressed during design. High priorities must be addressed during implementation.

6 System Design

1. What is system design?
   The process of desinging elements of a system (so system design is designing systems lol).

2. What are 3 of the 8 parts of system design.
   Design goals
   System decomposition
   Concurrency
   Hardware/Software Mapping
   Data management
   Global Resource Handling
   Software Control
   Boundary Conditions

3. Give two examples of design trade offs.
   Functionality vs Usability
   Efficency vs Portability
   Rapid Develppment vs Functionality
   Cost vs Reusability
   Cost vs robustness
   Backward compatability vs readability

4. Out of one of the two explain further why.
   The correctness is up to writers verification. I literally cannot write hundreds of explainations as to why in various different niche circumstances. You're probably correct however.

5. What is a Service subsystem?
   A group of operations provided by the subsystem.

6. How is a service specified?
   With the subsystem interface. Specifies interactions and information flow from/to subsystem boundaries. Should be well-defined and small. Often called an API but bad term during system design.

7. Criteria for subsystem selection should be where?
   Within subsystems we strive for high cohesion.

8. What is cohesion?
   Dependence among classes within a singular subsystem (this is poggers).

9. Whats the difference between high and low cohesion?
   In low cohesion there is a lot of miscellaneous classes with few assocations. High cohesion there are classes that perform similar tasks and are relation to eachother (assocations!).

10. What is coupling?
    Depedence between sub systems.

11. should you priotize cohesion or coupling.
    Cohesion should be strived for while coupling should be avoided if possible.

12. What problem does high coupling bring.
    Changing things within one subsystem will cause problems on other systems.

13. What techniques can achieve high cohesion and low coupling.
    Partitioning and layers.

14. What is a partition?
    Vertically divide a system into several indepedent subsystems that provide services on the same level of abstraction.

15. What is a layer?
    A subsystem that provides services to a higher layer (hierarchies).

16. What is closed architecture?
    Any layer is only able to invoke oeprations from the immediate layers below.

17. why bother with closed architecture?
    Highly maintainable.

18. What is open architecture.
    Any layer is only able to invoke operations from any layer below (Literally Hamdy's contour diagram).

19. Why bother with open architecture?
    Runtime Efficency

20. What is system decomposition?
    Identification of subsystems, servies, and their relationshp to eachother.

21. What are some types of software architecture?
    Pipe and filter
    Repositor
    Client/server
    Peer-To-Peer
    MVC (Model view controller)
    Layers

22. What is a pipe and filter?
    Input data originates from data source components. Output data is consumed by data sink components. A series of filters are put between linked by pipes.

23. What does a filter do?
    It is a transformation required to convert the input into the output.

24. When to use a pipe and filter?
    systems that apply transformations to streams of data without intervention of users.

25. What is the repository style?
    Subsystems access and modify data from a single data structure called a central repository. subsystems are loosely coupled and only interact through the repository

26. When should you use one?
    Applications with constantly changing, complex data processing tasks.

27. What is a problem with this style?
    It tends to have a bottle neck.

28. What is the client server style?
    One or many servers provide servies to instances of subsystems (clients).

29. T/F The server calls upon clients?
    False the client calls upon the server. It'd be a bit weird to have a server constantly asking if you're ready. (aye boss are you ready to turn off the lights yet?)

30. When should you use one?
    Typically data base systems.

31. What are the goals of client server style?
    Service portability: A server can be installed on a variety of machines and OS's in a variety of networking enviroments.
    Transparency The server might itself but distributed but should provide a single logical service to the user.
    Perforamnce client should provide interactive dispaly intensive tasks while the server does the CPU intensive operations.
    Scalability Server should have the capacity to handle a large number of clients.
    Reliability system should service node or communcation link problems.

32. What is the Peer-To-Peer architecture styel?
    Generalization of client server however now everyone can be a server or a client.

33. What are the three types of subsystems in MVC?
    Model, view, controller.

34. What is MVC?
    A special case of repository architecture. Model subsystems implements central data structures, controller subsystem explicitly dicates the control flow.

35. What is a microservice?
    Where software is decomposed into small indepedent systems that all work together.

36. What is monolith architecture?
    When a singular system where all services are within the same system.

37. What is the benefit of a microservice?
    Deployement flexibility, techonlogy flexibiilty, and can be scaled seperately.

38. what are some disadvantage?
    Deployement complexity and service discovery.

7 Testing

1. What is the cost of testing?
   At least half of your development budget will be spent on testing.

2. Is not testing a great way of saving money?
   No, infact it can actually cost more money. Debugging later and later into a project tends to cost more due to the 1-10-100 rule.

3. What are the four types of activities testing can be broken up into and what type of knowledge is required?
   Test design: Designing test values to satisfy coverage criteria or other engineering goals. Requires knowledge of discrete math, programming, and testing.
   Test automation: Embed test values into executable scripts. Requires knowledge of scripting.
   test execution: run tests on software and record the results. Requires little knowledge.
   Test evaluation: Evaluate results of testing and report to developers. Requires domain knowledge.

4. What is validation in testing?
   The process of evaluating software at the end of software development to esnure compliance with intended usage.

5. what is verification in testing?
   The process of determing whether the products of a given phase or software development process fulfills the requirement established during the previous phase.

6. What is a failure in testing.
   Any deviation of the observed behavior from the specified behavior.

7. what is an error in testing?
   The system in a state such that further processing by the system will cause a failure.

8. What is a fault in testing?
   The mechanical or algorithmic cause of an error. Also known as a bug.

9. what are some ways of dealing with errors? verification, Modular redundancy, Declaring a bug to be a feature, Patching, and testing.

10. What is testing? Finding inputs that cause the software to fail.

11. What is debugging? The process of finding a fault given a failure.

12. what is a test case? A set of test inputs, execution conditions, and expected results. Test inputs are the values that directly satisfy one test requirement.

13. What is black box testing?
    Focusing on I/O behavior. If we are able to predict the output for any given input. Typically impossible to generate all possible inputs.

14. What is the goal of black box testing?
    Reduce the number of test cases by equivalence partitioning. Divide input conditions into equivalnce classes and choose a representative for each class.
    equivalnce classes are determined through coverage and being disjoint.

15. What is white box testing?
    Testing with the goal of Thoroughness/coverage. Every statement in the component is executed at least once.

16. What are the types of White-box testing? Statement testing, Loop testing, Path Testing, Branch Testing.

17. What is static Analysis?
    Analysis done without running a program.

18. what are the types of static Analysis?
    Hand execution (reading the source code), Walk-Through (information presentation), Code inspection (Formal presentation), Automated tools for checking syntatic and semantic errors or a departure from coding standards.

19. What id dynamic Analysis?
    Testing done with running a program.

20. What are the types of dynamic analysis?
    Black-Box testing and White-box testing.

21. Is it possible to completely test any abritary system?
    No, you'd have to solve the halting problem and even if it was solved it'd cost a lot of time and money.

22. What are the 4 testing steps?
    Select what needs to be measured, Decide how the testing is done, develop test cases, and create a test oracle.

23. What is a test oracle?
    The set of predicted results for a set of test cases. It must be written down before the testing occurs.

24. What is unit testing?
    The testing of a unit. (Quite helpful I guess...)

25. What is a unit? A module or a small set of modules. A few examples, classes or interfaces.

26. Why bother with unit testing?
    It's practical and allows a divide-and-conquer approach. Splitting systems into units is very helpful and narrow down where bugs are. You don't want to chase down bugs in other units. Support regresion testing allows you to make changes to code and know if you broke something. Improves confidence that changing one many things doesn't break everything (yay!).

27. How can one do unit testing?
    Build systems in layers. Then test upwards. Start with classes that don't depend on others and countinue testing building on already tested classes.

28. Benefits of unit testing? Avoid having to write stubs. When testing a module it depends on verified reliable modules.

29. What are some unit testing Heurisitcs?
    Create unit tests as soon as object design is completed. Develop the test cases. Cross-check the test cases to eliminate duplicats. Desk check your source code. Create a test harness. Execute the test cases. Compare the results of the test with your orcale.

30. What are the parts of test code?
    The test fixture, test driver, and test oracle.

31. What is the test driver?
    The class that runs the tests.

32. What is the test fixture?
    Set of variables used in testing.

33. What is the intergration testing strategy?
    The entire system is viewed as a collection of subsystems. Note: The order which subsystems are selected for testing and intergration determines the specific strategy.

34. What is system testing?
    Ensure that the complete system compiles with the functional and nonfunctional requirements. Functional testing, Perforamnce testing, Acceptance testing, and installation testing are all different types of system testing.

35. What is the impact of requirements on system testing?
    The more explicit the measurements the easier they are to test. Quality of use cases determines ease of functional testing. Quality of subsystem determines the ease of structure testing. Quality of nonfunctional requirements and constraints determine the ease of perforamnce tests.

36. What are test requirements?
    Specific thigns that must be satisfied or covered during testing.

37. What are test criterion?
    A collection of rules and a process that defines test requirements.

38. What is another way of describing coverage in testing. (Hint think of test sets).
    Given a set of test requirements X for coverage criterion C, a test set T satisfies C coverage iff for every test requirement in X there is at least one test in T it is satisfied by a test requirement.

39. What is criteria subsumption? A test criterion C1 subsums C2 iff every set of test cases that satisfies criterion C1 also satisfies C2. (basically subsets)

40. what are some criteria structures?
    Graphs, Logical expresions, Input domain characterization, syntatic structures.

41. What is predicate coveage in Logical expresions?
    Each predicate must be True or False.

42. What is closure Coverage in Logical expresions?
    Each clause much be true or false.

43. Combinatorial coverage in Logical expresions?
    Various combinations of clauses.

44. What is active caluse coverage?
    Each caluse must determine the predicate's result.

45. What is input domain characterization?
    They describe the input domain of the software. Identify inputs, parameters, or other categorization. Partition each input into finite stes of representative classes. Choose combinations of values.

46. What are syntatic strcutres?
    Based on a grammar or other syntactic definition.

47. What is mutation testing and what is it an example of?
    When you introduce small changes tot he program and find tests that cause the mutant programs to fail. Failure is determined by a different output from the original program. Check the output of useful tests on the original program.

48. What is Model-Based testing.
    Derives tests from a model that describes some aspects of the system under a test. Model describes part of the behavior.

49. What is a coveage graph?
    The most commonly used structure for testing. Made up of graphs where tests are intended to cover the graph in some way.

50. What is a graph? A nonempty set Z of nodes. A set nonempty set X of initial nodes. A nonempty set of final nodes Y. A set E of edges where each edge connects one node to another.

51. What is a path in a graph?
    A sequence of nodes.

52. What is the length in a graph?
    The number of edges.

53. what is a subpath?
    A subsequence of nodes in p is a subpath of p.

54. What is reach in graphs?
    subgraph that can be reached from a node n.

55. What is a test path?
    A path that starts at an initial node and ends at a final node. Test paths represent execution of test cases.

56. what is an SESE graph?
    All test paths start at a single node and end at another node. Single-entry and single-exit.

57. What is visiting?
    A test path p visist a node if a node n is in p.

58. What is touring?
    A test path p tours a subpath q if q is a subpath of p.

59. what is Node coveage?
    Requires that each node and edge in a graph be executed. Test set T satisfies node coveage on a graph G iff for every syntatically reachable node n in N, there is some path in path(T) such that p visits n. (simple version Test requirements contain each reachable node in G.)

60. What is edge coverage?
A slightly stronger system than node coverage. Test requirements contain each reachable path of length up to 1, inclusive, in G.
Length up to 1 allows graphs with one and no edges.

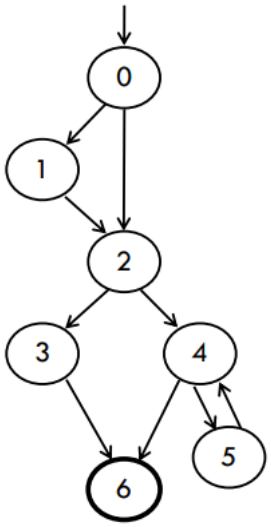
61. When are Node coveage and edge coverage difference?
When there is an edge and another subpath between a pair of nodes.
Node coverage TR = $\{0, 1, 2\}$.
Test path = $[0, 1, 2]$.
Edge coverage TR = $\{0, 1, 2\}$.
Test path = $[0, 1, 2], [0, 2]$.

62. What is edge pair coveage?
Requires pairs of edges or subpaths of length 2. Test requirements contain each reachable path of length up to 2, inclusive, in G.

63. What is complete path coverage and where does it fail?
Test requirements contain all paths in G. It fails whenever the program introduces a loop. Infiniately many TR paths arrise unfornately (might be solvable with the infinite monkey theorem assuming you have infinite monkeys...).

64. Specified Path coverage is?
Test requirements contain a set S of test paths, where S is supplied as a parameter.

65. The following questions use the following graph.



66. What are the test paths of the following image?

Node Coverage

TR = { 0, 1, 2, 3, 4, 5, 6 }

Test Paths: [0, 1, 2, 3, 6] [0, 1, 2, 4, 5, 4, 6]

67. What are the test paths of the following image?

Edge Coverage

TR = { (0,1), (0,2), (1,2), (2,3), (2,4), (3,6), (4,5), (4,6), (5,4) }

Test Paths: [0, 1, 2, 3, 6] [0, 2, 4, 5, 4, 6]

68. What are the test paths of the following image?

Edge-Pair Coverage

TR = { [0,1,2], [0,2,3], [0,2,4], [1,2,3], [1,2,4], [2,3,6], [2,4,5], [2,4,6], [4,5,4], [5,4,5], [5,4,6] }

Test Paths: [0, 1, 2, 3, 6] [0, 1, 2, 4, 6] [0, 2, 3, 6] [0, 2, 4, 5, 4, 5, 4, 6]

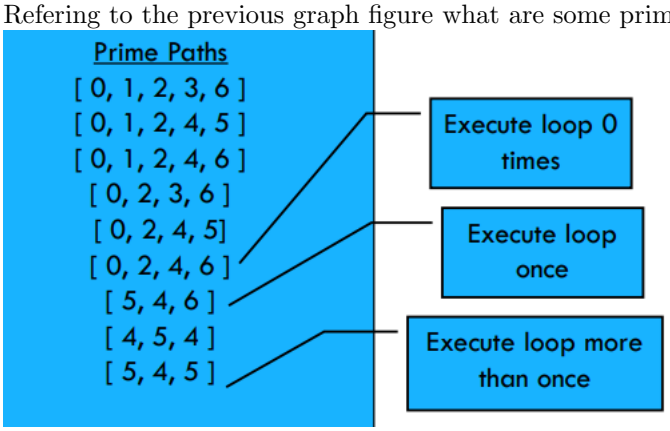
69. What are the test paths for Complete Path Coverage?
infinitely many paths exist unfornately... As long as a path reaches the end it'd be considered

70. What is a simple path?
If a given path from two nodes no node appears more than once except maybe the first and last nodes are the same.
There are no internal loops, include all other subpaths, a simple path.

71. What is a prime path?
A simple path that does not appear as a proper subpath of any other simple path.

72. What is prime path coverage?
A simple, elegant, and finite criterion that requires loops to be executed as well as skipped. Test requirements contain each prime path in G.
Will tour all paths of any lenghth. Subsumming node, edge, and edge-pair coverage.

73. Refering to the previous graph figure what are some prime paths?



74. What is data flow criteria.
A way to ensure that values are computed and used correctly.

75. What are the parts of Data flow criteria?
    Definition (def) the location where a value for a variable is stored into memory. Use a location where a variable's value is accessed. def (n) or def (e) the set of variables defined by node n or edge e. use (n) or use (e) the set of variables used by node n or edge e.

76. What is a DU pair?
    A pair of locations (x, y) such that a variable v is defined at x and used at y.

77. What is Reach in a DU pair?
    If there is a def-clear path from X to Y with respect to v, the def of v at x reaches the use at y.

78. what is Def-Clear?
    A path with respect to variable v if v is not given another value on any of the nodes/edges in the path. No redefinitions.

79. What is a du-path? A simple subpath that is def-clear with respect to v from def to use.

80. What is graph coverage for source code?
    The most common application is to the graph source. Made of graphs usually the control flow graph, Node coverage execute every statement, edge coverage execute every brach, loops looping structures, and data flow coverage augment the control flow graph.

81. What is a control flow graph.
    Modles all executions of a method by describing control structures. Made of nodes statements or sequences of statements, Edges transfers of control, and basic block a sequence of statements such that if the first statement, all statements will be (branchless). Typically annotated with extra information.