1. 소프트웨어 위기(crisis)에 대한 설명으로 가장 옳지 않은 것은?
① 소프트웨어 생산성이 사용자들의 서비스에 대한 요구를 따라가지 못한다.
② 소프트웨어 품질 향상과 유지보수가 힘들다.
③ 대부분의 경우 프로젝트 개발 일정과 소요 비용 예측이 부정확하다.
④ 고객의 기대치를 낮추고, 개발 생산성은 높여 해결하는 것이 바람직하다.

2. <보기>와 같은 장점을 가지는 테스팅 기법은?

<보기>
- 테스팅이 매일 시행되기 때문에, 비호환성 및 다른 중대한 오류들을 조기에 발견할 수 있어서 통합 위험이 최소화된다.
- 접근 방식이 통합 지향적이기 때문에, 컴포넌트 수준의 설계 오류뿐만 아니라 기능 오류를 발견하기 쉬우므로 최종 제품의 품질이 향상된다.
- 테스팅 중에 발견된 오류는 '소프트웨어의 새로운 증분(increment)'과 관련된 경우가 많다. 즉, 빌드에 새롭게 추가된 소프트웨어가 새로 발견된 오류의 원인일 가능성이 크기 때문에 오류 진단과 수정이 간단해진다.
- 날이 갈수록, 더 많은 소프트웨어가 통합되어 잘 동작하는 것을 보게 된다. 이러한 상황은 팀의 사기를 높이며 관리자들은 프로젝트 진척도에 대해 잘 알게 된다.

① 베타 테스팅 ② 스모크 테스팅
③ 회귀 테스팅 ④ 복구 테스팅

3. 구조적 방법론에서 모델링에 사용하는 표현 도구인 자료 흐름도(DFD: Data Flow Diagram)의 구성 요소가 아닌 것은?
① 자료 사전(Data Dictionary)
② 프로세스(Process)
③ 자료 저장소(Data Store)
④ 자료 흐름(Data Flow)

4. 비기능적 요구사항에 대한 설명으로 가장 옳지 않은 것은?
① 시스템에서 사용하는 최대 메모리는 1GB를 넘지 않아야 한다.
② 시스템은 초당 100개 이상의 트랜잭션을 처리할 수 있어야 한다.
③ 시스템의 가용 상태는 99.0% 이상이어야 하며 다운 시간이 1분을 초과할 수 없다.
④ 시스템의 현재 상황에 대한 음성 안내 기능을 제공해야 한다.

5. PMBOK(프로젝트관리지식체계)의 9가지 관점 중 프로젝트 통합관리에 해당하지 않는 것은?
① 프로젝트 계획 개발 ② 일정 통제
③ 프로젝트 계획 실행 ④ 통합된 변경 통제

6. <보기>의 클래스 다이어그램을 작성하는 과정을 순서대로 바르게 나열한 것은?

<보기>
ㄱ. 가장 확실한 일반화 관계부터 시작하여 파악한다.
ㄴ. 클래스의 주요 임무를 찾아 나열한다.
ㄷ. 클래스 후보 집합을 파악한다.
ㄹ. 가장 중요한 클래스부터 시작하여 연관 관계와 꼭 필요한 속성을 추가한다.
ㅁ. 임무를 기준으로 필요한 오퍼레이션을 결정한다.

① ㄱ－ㄴ－ㄷ－ㄹ－ㅁ ② ㄱ－ㄹ－ㅁ－ㄴ－ㄷ
③ ㄷ－ㄴ－ㄱ－ㅁ－ㄹ ④ ㄷ－ㄹ－ㄱ－ㄴ－ㅁ

7. 맥콜(McCall)의 품질 요소 분류 중 제품 개선(product revision)에 해당하지 않는 것은?
① 유지보수 용이성(maintainability)
② 정확성(correctness)
③ 테스트 용이성(testability)
④ 유연성(flexibility)

8. 소프트웨어 아키텍처를 잘 설계하기 위한 고려 사항에 대한 설명으로 가장 옳지 않은 것은?
① 시스템 품질 속성을 결정할 수 있어야 한다.
② 의사소통 도구로 활용할 수 있어야 한다.
③ 구현에 대한 제약 사항은 고려하지 않아도 된다.
④ 재사용할 수 있게 설계해야 한다.

9. 소프트웨어에 대한 설명으로 가장 옳지 않은 것은?
① 소프트웨어는 제조되는 것이 아니고, 개발되는 것이다.
② 소프트웨어란 프로그램과 프로그램의 개발, 운용, 보수에 필요한 모든 관련 정보를 말한다.
③ 소프트웨어는 수학이나 물리학처럼 규칙적이고 정형적인 구조를 가진다.
④ 소프트웨어는 그 생산물의 구조가 코드 안에 숨겨져 있는 비가시성(invisibility)을 갖는다.

10. LOC 기법에 의해 예측된 총 라인 수가 60,000라인이고, 개발자는 5명이 참여한다. 개발자들이 1인당 월 평균 600라인을 코딩할 때, 개발 기간은?
① 10개월 ② 15개월 ③ 20개월 ④ 25개월

11. 객체지향 소프트웨어 개발을 위한 UP(Unified Process)의 개발 단계에 속하지 않는 것은?
① 도입 단계(inception phase)
② 구축 단계(construction phase)
③ 검증 단계(validation phase)
④ 전이 단계(transition phase)

12. <보기>는 소프트웨어 공학 계층에 대한 설명이다. ㈎~㈐의 설명에 해당하는 용어를 옳게 짝지은 것은?

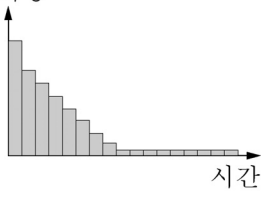
<보기>
㈎ 기술 계층을 묶어주는 접착제 역할을 하는 것이고 소프트웨어를 합리적으로 빨리 개발할 수 있도록 한다.
㈏ 소프트웨어를 개발하기 위한 전문적인 실용기술을 제공한다.
㈐ 소프트웨어 개발의 각 계층에 사용된 개념들에 대해 자동적 또는 반자동적 지원을 한다.

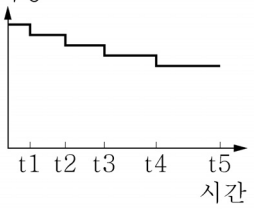
	㈎	㈏	㈐
①	방법(methods)	프로세스(process)	도구(tool)
②	도구(tool)	방법(methods)	프로세스(process)
③	프로세스(process)	도구(tool)	방법(methods)
④	프로세스(process)	방법(methods)	도구(tool)

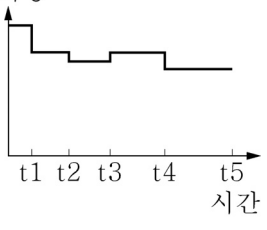
13. PERT/CPM에 대한 설명으로 가장 옳지 않은 것은?
① 임계 경로는 그래프에서 작업 진행에 여유 시간이 없는 경로를 말한다.
② 전체 프로젝트 완료 시간을 단축하고 싶다면 임계 경로의 작업시간을 줄여야 한다.
③ 병행 작업을 계획할 수 있다.
④ 그래프에서 가능한 전체 작업 시간의 합이 가장 작은 경로의 작업들을 잘 관리하여야 한다.

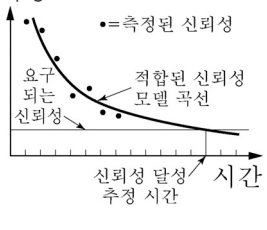
14. 그래프에서 신뢰성(ROCOF)은 소프트웨어 결함 발생 비율을 의미한다. <보기>에서 설명하는 모델에 해당하는 그래프는?

<보기>
• 소프트웨어 결함을 수정함에 따라 동일한 정도로 신뢰성이 향상되지 않고 랜덤 분포에 따라 변화한다.
• 디버깅 동안 소프트웨어 결함이 항상 수정되지는 않고, 때로는 새로운 결함이 삽입된다.
• 결함의 수정에 따라 새로운 결함이 야기되면 신뢰성이 감소한다.
• 자주 발생하는 결함은 테스팅 프로세스 초기에 발견되는 경향이 있기 때문에, 결함이 수정됨에 따라 평균적인 신뢰성 개선 정도가 감소된다.

① 신뢰성


② 신뢰성


③ 신뢰성


④ 신뢰성


15. <보기>에서 설명하는 디자인 패턴은?

<보기>
• 객체 생성 인터페이스를 정의하기 위한 디자인 패턴이다.
• 상위클래스에서는 객체를 만드는 인터페이스를 정의하고, 하위클래스에서는 구체적인 인스턴스를 생성하도록 한다.
• 객체를 생성하는 인터페이스와 실제 객체를 생성하는 클래스를 분리할 수 있다.

① factory method 패턴 ② prototype 패턴
③ builder 패턴 ④ abstraction factory 패턴

16. 소프트웨어의 설계에서 모듈 사이의 좋은 관계로 가장 옳지 않은 것은?
① 모듈 간에는 꼭 필요한 데이터만 주고받는 것이 좋다.
② 약한 결합을 유지하기 위해 인터페이스가 복잡하지 않아야 한다.
③ 가장 좋은 방법은 모듈 간의 결합도는 낮게, 응집도는 높게 하는 것이다.
④ 모듈 간의 인터페이스는 데이터보다 제어 플래그를 사용하는 것이 좋다.

17. 블랙 박스 테스트에서 발견할 수 없는 에러는?
① 인터페이스 에러
② 제어구조 에러
③ 외부 데이터베이스 접근 에러
④ 틀리거나 누락된 기능

18. 요구사항 분석 모델 중 유스케이스(use case)가 속한 모델은?
① 시나리오 기반 모델(scenario based model)
② 클래스 모델(class model)
③ 행위 모델(behavioral model)
④ 플로우 모델(flow model)

19. <보기>와 같은 프로세스 영역들이 속하는 CMMI의 Maturity Level은?

<보기>
• 요구사항 개발(requirement development)
• 제품 통합(product integration)
• 통합된 프로젝트 관리(integrated project management)
• 리스크 관리(risk management)

① Level 1 ② Level 2
③ Level 3 ④ Level 4

20. 소프트웨어 아키텍처 유형 중 파이프 필터(pipe and filter) 구조가 속하는 것은?
① 데이터 중심 아키텍처(data-centered architecture)
② 데이터 흐름 아키텍처(data-flow architecture)
③ 계층 아키텍처(layered architecture)
④ 호출과 반환 아키텍처(call and return architecture)