소프트웨어공학 핵심 문제 & 풀이

* 출제빈도가 높은 문제만 정리했습니다.
문제, 정답, 해설을 함께 보면서 시험장 가기전 최종 마무리용으로 활용하세요.

[기-08년9월][기-00년3월][기-04년3월]
1. 소프트웨어 공학의 기본 원칙이라고 볼 수 없는 것은?
가. 현대적인 프로그래밍 기술 적용
나. 지속적인 검증 시행
다. 결과에 대한 명확한 기록 유지
라. 충분한 인력 투입

=> 문제에서 아니다 싶은 답이 있을 겁니다.
소프트웨어 공학은 생산성과 연관이 있습니다. 최소 인력을 투입해서 품질 좋은 S/W 를 만들어야 됩니다.

[기-04년9월][기-07년9월]
2. 소프트웨어 개발의 생산성에 영향을 미치는 요소가 아닌 것은?
가. 프로그래머의 능력 나. 팀 의사 전달
다. 제품의 복잡도 라. 소프트웨어 사용자의 능력

=> 문제에서 아니다 싶은 답이 있을 겁니다.
개발의 생산성과 사용자는 거리가 있습니다.

[기-06년9월][기-05년3월]
3. 소프트웨어 공학의 발전을 위한 소프트웨어 사용자(Software User)로서의 자세로 옳지 않은 것은?
가. 프로그래밍 언어와 알고리즘의 최근 동향을 주기적으로 파악한다.
나. 컴퓨터의 이용 효율이나 워크스테이션에 관한 정보들을 체계적으로 데이터베이스화 한다.
다. 타 기업의 시스템에 몰래 접속하여 새로운 소프트웨어 개발에 관한 정보를 획득한다.
라. 바이러스에 대한 예방에 만전을 기하여 시스템의 안전을 확보한다.

=> 다번은 해킹

[기-08년3월][기-07년5월][기-02년5월][기-99년8월]
4. 소프트웨어의 위기현상과 거리가 먼 것은?
가. 유지보수의 어려움
나. 개발인력의 급증
다. 성능 및 신뢰성의 부족
라. 개발기간의 지연 및 개발비용의 증가

=> 개발인력의 부족 -> 인건비 상승 -> 개발기간 지연 및 개발 비용 증가

[기-06년5월][기-02년9월]
5. 컴퓨터의 발달 과정에서 소프트웨어의 개발 속도가 하드웨어의 개발 속도를 따라가지 못해 사용자들의 요구 사항을 감당할 수 없는 문제가 발생함을 의미하는 것은?
가. 소프트웨어 위기(Crisis)
나. 소프트웨어 오류(Error)
다. 소프트웨어 버그(Bug)
라. 소프트웨어 유지보수(Maintenance)

[기-06년5월][기-05년3월]
6. 소프트웨어 공학에 대한 가장 적절한 설명은?
가. 소프트웨어 위기(software crisis)를 완전히 해결한 공학적 원리의 체계이다.
나. 신뢰성 있는 소프트웨어를 만들기 위한 도구만을 연구하는 학문이다.
다. 가장 경제적으로 신뢰도 높은 소프트웨어를 만들기 위한 방법, 도구와 절차들의 체계이다.
라. 점차 많은 비용이 소요되는 소프트웨어 개발에서 가장 경제적인 방법을 찾고자 하는 것이다.

[기-06년9월][기-01년9월]
7. 다음 중 전통적인 소프트웨어 개발 방법론인 폭포수형(waterfall) 모델에서 개발 순서가 옳은 것은?
가. 타당성 컴토 → 계획 → 분석 → 구현 → 설계
나. 타당성 컴토 → 분석 → 계획 → 설계 → 구현
다. 타당성 컴토 → 계획 → 분석 → 설계 → 구현
라. 타당성 컴토 → 분석 → 계획 → 구현 → 설계

=> 계획 다음에 요구분석입니다. 주의하세요.

[기-03년8월][기-99년4월]
8. 소프트웨어 수명주기 모형 중 폭포수 모형에 대한 설명으로 옳지 않은 것은?
가. 적용사례가 많다.
나. 단계별 정의가 분명하다.
다. 단계별 산출물이 명확하다.
라. 요구사항의 변경이 용이하다.

=> 순차적 진행 -> 개발 과정 중에 발생하는 새로운 요구나 경험을 설계에 반영하기 어려움

[기-03년3월][기-01년3월]
9. 폭포수 모형(waterfall model)의 진행 단계로 옳은 것은?

① 시험	② 분석	③ 계획
④ 코딩	⑤ 유지보수	⑥ 설계

가. ①-②-③-④-⑤-⑥ 나. ②-⑥-④-⑤-①-③
다. ③-②-⑥-④-①-⑤ 라. ④-①-②-⑥-⑤-③

=> 계획 다음에 요구분석입니다. 주의하세요.

[기-02년9월][기-99년8월]
10. 폭포수 모델(Waterfall model) 에 대한 설명으로 옳지 않은 것은?
가. 앞 단계가 끝나야만 다음 단계로 넘어갈 수 있다.
나. 요구분석 단계에서 프로토타입을 사용하는 것이 특징이다.
다. 제품의 일부가 될 매뉴얼을 작성해야 한다.
라. 각 단계가 끝난 후 결과물이 명확히 나와야 한다.

[기-07년3월][기-05년9월]
11. 사용자의 요구사항 분석 작업이 어려운 이유와 거리가 먼 것은?
가. 개발자와 사용자 간의 지식이나 표현의 차이가 커서 상호 이해가 쉽지 않다.

나. 사용자의 요구는 예외가 거의 없어 열거와 구조화가 어렵지 않다.
다. 사용자의 요구사항이 모호하고 부정확하며, 불완전하다.
라. 개발하고자 하는 시스템 자체가 복잡하다.

=> 실세계는 예측할 수 없는 예외가 존재하므로 개발하기 쉽지 않다.

[기-06년9월][기-04년9월][기-99년8월][기-01년6월]
12. 프로토타이핑 모형(Prototyping Model)에 대한 설명으로 옳지 않은 것은?
가. 최종 결과물의 만들어지기 전에 의뢰자가 최종 결과물의 일부 또는 모형을 볼 수 있다.
나. 개발 단계에서 오류 수정이 불가능하므로 유지보수 비용이 많이 발생한다.
다. 프로토타입은 발주자나 개발자 모두에게 공동의 참조 모델을 제공한다.
라. 프로토타입은 구현 단계의 구현 골격이 될 수 있다.

[기-06년5월][기-06년3월][기-05년3월][기-03년3월][기-00년7월][기-07년9월]
13. 프로토타입 모형의 장점으로 가장 적절한 것은?
가. 비용과 시간의 절감 나. 책임 한계의 명백한 구분
다. 요구사항의 충실 반영 라. 프로젝트 관리의 용이

=> 프로토타입(모델하우스) : 요구사항을 충실히 반영

[기-05년9월][기-00년3월]
14. 시스템의 일부 혹은 시스템의 모형을 만드는 과정으로서 요구된 소프트웨어의 일부를 구현하며, 추후 구현단계에 사용될 골격코드가 되는 모형은?
가. 폭포수 모형 나. 점증적 모형
다. 프로토타이핑 모형 라. 계획수립 모형

[기-08년3월][기-05년3월][기-01년9월]
15. 소프트웨어 수명주기 모형 중 프로토타이핑 모형(prototyping model)의 가장 큰 장점은?
가. 위험요소가 쉽게 발견된다.
나. 유지보수가 쉬워진다.
다. 사용자 요구사항을 정확하게 파악할 수 있다.
라. 소프트웨어 개발 일정을 정확하게 수립할 수 있다.

[기-04년3월][기-01년3월]
16. 실제 상황이 나오기 전에 가상으로 시뮬레이션을 통해 최종 결과물에 대한 예측을 할 수 있는 소프트웨어 수명 주기 모형은?
가. 점증적 모형(spiral model)
나. 프로토타이핑 모형(prototyping model)
다. 코코모 모형(cocomo model)
라. 폭포수 모형(waterfall model)

[기-04년9월][기-05년5월][기-99년8월]
17. 소프트웨어 수명주기 모형 중 나선형(spiral) 모델의 처리 절차에 해당되지 않는 것은?
가. 계획수립 나. 고객 평가
다. 위험 분석 라. 시스템 유지보수

=> 계획수립 -> 위험분석 -> 공학적개발 -> 고객평가

[기-03년5월][기-02년3월]
18. Boehm 이 제안한 나선형 모델의 태스크(task)에 해당되지 않는 것은?
가. 계획 수립(Planning)
나. 위험 분석(Risk Analysis)
다. 객체 구현(Object Implementation)
라. 고객 평가(Customer Evaluation)

[기-01년9월][기-04년3월]
19. 프로젝트 관리의 대상으로 거리가 먼 것은?
가. 비용관리 나. 일정관리 다. 고객관리 라. 품질관리

[기-08년3월][기-00년10월][기-07년3월][기-00년7월][기-99년8월][기-02년3월][기-05년3월][기-05년5월][기-06년5월][기-06년9월][기-07년9월]
20. 효과적인 소프트웨어 프로젝트 관리를 위한 3P에 해당되지 않는 것은?
가. people(사람) : 인적자원
나. product(생산물) : 생산일정
다. problem(문제) : 문제인식
라. process(프로세스) : 작업계획

=> 사람(people)은 절차(process)에 맞도록 문제(problem)를 해결해야 한다.

[기-99년4월][기-03년5월]
21. 소프트웨어 프로젝트 계획 수립시 소프트웨어 영역(Software Scope) 결정 사항에 포함되지 않는 것은?
가. 기능 나. 성능 다. 위험성 라. 제약조건

=> 기능, 성능, 제한조건, 신뢰도, 위험성 (X)

[기-00년10월][기-99년8월]
22. S/W project를 계획하는 데 있어 예측의 대상으로 거리가 먼 것은?
가. 비용 나. 일정 다. 성능 라. 위험요인

=> 위험요인(돌방상황)은 예측이 어렵죠.

[기-99년4월][기-02년9월]
23. 중앙집중식 팀 구성에 관한 설명 중 틀린 것은?
가. 소프트웨어 개발팀을 중앙집중형으로 관리하는 방법에는 책임 프로그래머팀이 있다.
나. 프로그램 사서(Program Libraian)는 프로그램 리스트, 설계문서, 테스트 계획 등을 관리한다.
다. 중앙집중식 팀 구성은 한 사람에 의하여 통제할 수 있는 비교적 소규모 문제에 적합하다.
라. 보조 프로그래머는 요구분석과 설계, 중요한 부분의 프로그래밍 및 모든 기술적 판단을 내린다.

=> 책임 프로그래머가 모든 기술적 판단을 하고 나머지는 보조적이 업무만 한다.

[기-06년3월][기-07년5월]
24. 민주주의적 팀(Democratic teams)에 대한 내용으로 옳은 것은?
가. 프로젝트 팀의 목표 설정 및 의사 결정 권한이 팀 리더에게 주어진다.
나. 조직적으로 잘 구성된 중앙 집중식 구조이다.
다. 팀 구성원간의 의사 교류를 활성화 시키므로 팀원의 참여도와 만족도를 증대시킨다.

라. 팀 리더의 개인적 능력이 중요하다.

=> 모든 팀 구성원이 동등한 위치에서 의사 결정 -> 의사 교류를 활성화 -> 구성원의 작업 만족도 증대

[기-08년5월][기-08년3월][기-01년3월][기-06년3월]
25. LOC 기법에 의하여 예측된 총라인수가 25000 라인일 경우 개발에 투입될 프로그래머의 수가 5명이고, 프로그래머들의 평균 생산성이 월 당 500 라인일 때, 개발에 소요되는 시간은?
가. 8개월 나. 9개월 다. 10개월 라. 11개월

=> 한 달 동안 개발할 수 있는 라인수:5*500=2,500
25,000 라인 개발 소요 개월:25,000/2,500=10

[기-05년9월][기-99년10월]
26. 소프트웨어 비용을 정확하게 예측하기 위한 방법 중 그 현실성이 적은 것은?
가. 경험적 모형을 이용한다.
나. 분해기법을 이용한다.
다. 예측을 가능한 뒤로 미룬다.
라. 과거의 유사한 프로젝트를 이용한다.

[기-00년3월][기-03년5월]
27. 비용예측방법에서 원시 프로그램의 규모에 의한 방법(COCOMO model)중 최대형 규모의 트랜잭션 처리 시스템이나 운영체제 등의 소프트웨어를 개발하는 유형은?
가. Organic 프로젝트
나. Semi-detached 프로젝트
다. Embedded 프로젝트
라. Organic, Embedded 프로젝트

=> ① Organic 프로젝트 (유기형) : 5만 라인 이하 규모 (일괄처리, 과학 기술 계산용 등)
② Semi-Detached 프로젝트 (반분리형) : 30만 라인 이하 규모 (운영체제 등)
③ Embedded 프로젝트 (내장형) : 30만 라인 이상의 최대형 규모 (운영체제 등)

[기-00년10월][기-02년3월][기-04년5월]
28. COCOMO 법에 의한 소프트웨어 모형에 속하지 않는 것은?
가. Basic COCOMO 나. Putnam COCOMO
다. intermediate COCOMO 라. Detailed COCOMO

=> ① Basic COCOMO (기본형)
② Intermediate COCOMO (중간형)
③ Detailed COCOMO (진보형)

[기-01년6월][기-03년3월]
29. COCOMO model 중 기관 내부에서 개발된 중소 규모의 소프트웨어로 일괄 자료 처리나 과학 기술 계산용, 비즈니스 자료 처리용으로 5만 라인 이하의 소프트웨어를 개발하는 유형?
가. semi-datached model 나. organic model
다. semi-embedded model 라. embedded model

[기-01년9월][기-99년10월][기-04년9월]
30. COCOMO의 프로젝트 모드가 아닌 것은?
가. organic mode 나. semi-detached mode
다. medium mode 라. embedded mode

[기-08년3월][기-08년9월][기-01년9월][기-03년5월][기-02년9월][기-00년3월][기-06년9월][기-07년5월][기-05년3월][기-04년9월][기-03년8월][기-07년9월]
31. S/W Project 일정이 지연된다고 해서 Project 말기에 새로운 인원을 추가 투입하면 Project는 더욱 지연되게 된다고 주장하는 법칙은?
가. Putnam의 법칙 나. Mayer의 법칙
다. Brooks의 법칙 라. Boehm의 법칙

[기-00년7월][기-00년10월][기-05년9월]
32. 일정계획 방법에서 이용되는 PERT/CPM (Program-Evaluation and Review technique/Critical Path Method)이 제공하는 도구가 아닌 것은?
가. 프로젝트 개발기간을 결정하는 임계경로
나. 통계적 모델을 적용해서 개별작업의 가장 근접한 시간 측정 기준
다. 정의작업에 대한 시작시간을 정의하여 작업들간의 경계시간 계산
라. 프로젝트 개발기간 중 투입되는 노력과 비용기준

=> 소요 기간을 예측하는 일정 계획 방법으로 비용 산출은 안된다.

[기-02년9월][기-05년9월]
33. CPM(Critical Path Method) 네트워크에 대한 설명으로 옳지 않은 것은?
가. 노드에서 작업을 표시하고 간선은 작업 사이의 전후 의존관계를 나타낸다.
나. 프로젝트의 완성에 필요한 작업을 나열하고 작업에 필요한 소요기간을 예측하는데 사용한다.
다. 박스노드는 프로젝트이 중간 점검을 뜻하는 이정표로 이 노드위에서 예상완료 시간을 표시한다.
라. 한 이정표에서 다른 이정표에 도달하기 전의 작업은 모두 완료되지 않아도 다음 작업을 진행 할 수 있다.

=> 한 이정표에서 다른 이정표에 도달하기 전의 작업은 모두 완료되어야 종합적인 정보를 이용해 다음 작업을 진행할 수 있다.

[기-99년8월][기-05년9월]
34. Gantt Chart에 포함되지 않는 사항은?
가. 이정표 나. 작업일정
다. 작업기간 라. 주요작업경로

=> 포함되는 내용 : 이정표, 작업 일정, 작업 기간, 산출문, 작업 경로 (X)

[기-08년9월][기-06년3월]
35. 프로젝트 일정 관리시 사용하는 간트(Gantt) 차트에 대한 설명으로 옳지 않은 것은?
가. 막대로 표시하며, 수평 막대 길이는 각 테스크의 기간을 나타낸다.
나. 이정표, 기간, 작업, 프로젝트 일정을 나타낸다.
다. 시간선(time-line) 차트라고도 한다.
라. 작업들 간의 상호 관련성, 결정 경로를 표시한다.

=> 간트 차트:작업 경로(X)

[기-01년3월][기-02년3월][기-05년3월][기-07년9월]

36. 소프트웨어 품질목표에 대한 설명으로 옳지 않은 것은?
가. 신뢰성(reliability) ; 정확하고 일관된 결과를 얻기 위해 요구된 기능을 수행하는 정도
나. 이식성(portability) : 다양한 하드웨어 환경에서도 운용 가능하도록 쉽게 수정될 수 있는 정도
다. 상호운용성(intercoperability) : 다른 소프트웨어와 정보를 교환할 수 있는 정도
라. 사용용이성(usability) : 전체나 일부 소프트웨어가 다른 응용 목적으로 사용될 수 있는 정도

=> 사용 용이성 : 사용하기 쉬운 정도, 재사용성 : 전체나 일부 소프트웨어가 다른 응용 목적으로 사용 가능

[기-01년6월][기-04년9월]
37. 다음 내용에 가장 적합한 것은?

"어떤 항목이나 제품의 설정된 기술적인 요구사항과 일치하는가를 적절하게 확인하는데 필요한 체계적이고도 계획적인 유형의 활동이다. "

가. 검열(inspections)
나. 품질보증(quality assurance)
다. 정적분석(static analysis)
라. 기호실행(symbolic execution)

=> 품질보증 : 요구사항과 일치하는가?

[기-01년9월][기-04년9월][기-07년5월]
38. 소프트웨어 품질관리 위원회의 기본적인 목적으로 가장 바람직한 것은?
가. 소프트웨어 품질 향상
나. 표준화 준수 여부 검증
다. 도큐먼트(document)의 품질 검사
라. 사용자와의 관계 향상

[기-08년9월][기-02년3월][기-99년10월][기-07년9월]
39. 소프트웨어 품질 보증을 위한 정형 기술 검토의 지침 사항으로 옳지 않은 것은?
가. 논쟁과 반박의 제한성 나. 의제의 무제한성
다. 제품검토의 집중성 라. 참가인원의 제한성

=> 의제의 제한이 필요하다.

[기-03년8월][기-06년5월]
40. 소프트웨어 품질관리 기술에서 품질목표의 항목과 거리가 먼 것은?
가. 정확성 나. 유지보수성
다. 무결성 라. S/W 종속성

=> 품질목표의 항목 오답:S/W 종속성,중복성,복잡성,최적화

[기-08년3월][기-05년3월][기-07년5월][기-06년9월]
41. 소프트웨어의 품질 목표 중에서 옳고 일관된 결과를 얻기 위하여 요구된 기능을 수행할 수 있는 정도를 나타낸 것은?
가. 정확성(correctness) 나. 신뢰성(reliability)
다. 효율성(efficiency) 라. 무결성(integrity)

=> 신뢰성:일관된 결과

[기-00년7월][기-02년3월][기-04년9월]
42. 위험성 추정을 위한 위험표(risk table)에 포함될 사항이 아닌 것은?
가. 위험 발생 시간 나. 위험 발생 확률
다. 위험의 내용 및 종류 라. 위험에 따르는 영향력

=> 위험성 추정이므로 발생시간을 포함할 수 없다.

[기-03년5월][기-07년5월]
43. 위험 모니터링(monitoring)의 의미로 가장 적절한 것은?
가. 위험을 이해하는 것
나. 위험요소들에 대하여 계획적으로 관리하는 것
다. 위험 요소 징후들에 대하여 계속적으로 인지하는 것
라. 첫번째 조치로 위험을 피할 수 있도록 하는 것

[기-05년3월][기-02년5월][기-06년5월]
44. 프로젝트 추진 과정에서 예상되는 각종 돌발 상황을 미리 예상하고 이에 대한 적절한 대책을 수립하는 일련의 활동을 무엇이라고 하는가?
가. 위험관리 나. 일정관리
다. 코드관리 라. 모형관리

=> 위험(돌방상황)

[기-01년3월][기-04년3월]
45. 형상관리(configuration management)의 관리 항목으로 거리가 먼 것은?
가. 정의 단계의 문서
나. 개발 단계의 문서와 프로그램
다. 유지보수 단계의 변경 사항
라. 소프트웨어 개발 비용

=> 형상관리 : 소프트웨어의 생산물(프로그램,문서,데이터)을 확인하고 소프트웨어 통제, 변경 상태를 기록하고 보관하는 일련의 작업 (유지보수 단계에서 행해진다.)

[기-08년3월][기-01년9월][기-03년8월]
46. 소프트웨어 형상관리(configuration management)에 관한 설명으로 가장 거리가 먼 것은?
가. 소프트웨어에서 일어나는 수정이나 변경을 알아내고 제어하는 것을 의미한다.
나. 유지보수 단계에서 행해진다.
다. 형상관리를 위하여 구성된 팀을 책임프로그래머 팀(chief programmer team)이라고 한다.
라. 형상관리에서 중요한 기술 중의 하나는 버전 제어 기술이다.

=> 책임프로그래머 팀은 개발시에 조직된 팀 구성 방법이다.

[기-08년5월][기-02년5월][기-04년5월][기-06년5월]
47. 소프트웨어에 대한 변경을 관리하기 위해 개발된 일련의 활동을 나타내며 이런 변경이 전체 비용이 최소화되고 최소한의 방해가 소프트웨어의 현 사용자에게 야기되도록 보증하는 것을 목적으로 하는 것은?
가. 위험 관리 나. 형상 관리
다. 프로젝트 관리 라. 유지보수 관리

=> 형상 관리 키워드 : 변경
위험 관리 : 프로젝트 추진 과정에서 예상되는 각종 돌발 상황(위험)을 미리 예상하고 이에 대한 적절한 대책을 수립하는

일련의 활동
프로젝트 관리 : 주어진 기간 내에 최소의 비용으로 사용자를 만족시키는 시스템을 개발하기 위한 전반적인 활동
유지보수 관리 : 개발된 소프트웨어의 품질을 향상 최상의 상태로 유지하기 위한 것으로, 소프트웨어 개발 단계 중 가장 많은 노력과 비용이 투입되는 단계

[기-05년3월][기-00년3월][기-02년9월][기-07년5월][기-06년9월]
48. 소프트웨어 형상관리(Software Configuration - Management)의 설명으로 가장 적합한 것은?
가. 소프트웨어 개발과정을 문서화하는 것이다.
나. 하나의 작업 산출물을 정해진 시간 내에 작성하도록 하는 관리이다.
다. 수행결과의 완전성을 점검하고 프로젝트의 성과 평가척도를 준비하는 작업이다.
라. 소프트웨어의 생산물을 확인하고 소프트웨어 통제, 변경 상태를 기록하고 보관하는 일련의 관리 작업이다.

=> 형상 관리 키워드 : 변경

[기-08년5월][기-03년8월][기-00년3월]
49. 구조적 분석 도구와 거리가 먼 것은?
가. 자료 사전 나. 자료 흐름도
다. 프로그램 명세서 라. 소단위 명세서

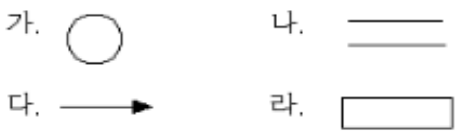
=> 구조적 분석 기법(도구)
- 자료의 흐름과 처리를 중심으로 하는 요구사항 분석 방법
- 종류 : 자료 흐름도, 자료 사전, 소단위 명세서, 개체 관계도, 상태 전이도
* 프로그램 명세서는 프로그램 설명서, 메뉴얼과 비슷합니다.

[기-05년3월][기-06년9월]
50. 시스템 개발을 위한 첫 단계는 사용자의 요구나 시스템에 대한 분석이라고 할 수 있다. 이 중 사용자의 요구 분석을 위해 주로 사용하는 기법이 아닌 것은?
가. 사용자 면접
나. 현재 사용 중인 각종 문서 검토
다. 설문 조사를 통한 의견 수렴
라. 통제 및 보안 분석

[기-99년10월][기-02년9월][기-07년5월]
51. 자료흐름도(DFD)의 구성요소가 아닌 것은?
가. 처리 나. 자료흐름 다. 단말 라. 기수

=> 직사각형(단말), 화살표(자료흐름), 원(처리), 직선,이중선(자료저장소)

[기-00년3월][기-01년9월][기-04년3월]
52. 자료흐름도(DFD:Data Flow Diagram)의 구성요소 중 자료출처와 도착지를 나타내는 기호는? 라



=> 직사각형(단말):자료의 출처,도착지

[기-01년3월][기-02년3월][기-05년9월][기-04년9월]
53. 자료흐름도에서 구성요소에 대한 기호의 표현 연결이 옳지 않은 것은?
가. 자료흐름 : 화살표로 표시
나. 처리공정 : 마름모로 표시
다. 자료저장장소 : 직선(단선, 이중선)으로 표시
라. 종착지 : 사각형으로 표시

[기-01년6월][기-03년5월]
54. 자료흐름도의 구성 요소와 표시 기호의 연결이 옳지 않은 것은?
가. 종착지(terminator) : 오각형
나. 자료흐름(data flow) : 화살표
다. 처리공정(process) : 원
라. 자료저장소(data store) : 직선

[기-08년9월][기-03년8월]
55. 자료흐름도의 구성요소가 아닌 것은?
가. 소단위명세서 나. 단말
다. 프로세스 라. 자료저장소

[기-06년9월][기-05년3월][기-07년9월]
56. 데이터 흐름도(DFD)의 구성요소에 포함되지 않는 것은?
가. 처리공정(process)
나. 자료흐름(data flow)
다. 자료사전(data dictionary)
라. 자료저장소(data store)

[기-08년9월][기-08년3월][기-06년9월][기-07년9월]
57. 자료 사전(Data Dictionary)에서 반복을 의미하는 것은?
가. = 나. { } 다. + 라. ( )

=> 정의 = , 반복 { } , 연결 + , 생략 ( )

[기-00년7월][기-02년3월][기-04년9월][기-06년9월]
58. HIPO(hierarchy plus input process output)에 대한 설명으로 옳지 않은 것은?
가. HIPO 다이어그램에는 가시적 도표(visual table of contents). 총체적 다이어그램(ovrview diagram). 세부적 다이어그램(detail diagram)의 세 종류가 있다.
나. 가시적 도표(visual table of contents)는 시스템에 있는 어떤 특별한 기능을 담당하는 부분의 입력, 처리, 출력에 대한 전반적인 정보를 제공한다.
다. HIPO 다이어그램은 분석 및 설계 도구로서 사용된다.
라. HIPO는 시스템의 설계나 시스템 문서화용으로 사용되고 있는 기법이며, 기본 시스템 모델은 입력, 처리, 출력으로 구성된다.

=> 총체적 다이어그램(Overview Diagram):입력,처리,출력

[기-01년3월][기-04년3월]
59. 프로그램을 구성하는 기능을 기술한 것으로 입력, 처리, 출력을 기술하는 HIPO 패키지에 해당하는 것은?
가. Overview Diagram 나. Detail Diagram
다. Visual Table of contents 라. Index Diagram

[기-02년9월][기-05년9월]
60. HIPO(Hierarchy Input Process Output)에 대한 설명으로 옳지 않은 것은?
가. 상향식 소프트웨어 개발을 위한 문서화 도구이다.

나. 구조도, 개요 도표 집합, 상세 도표 집합으로 구성된다.
다. 기능과 자료의 의존 관계를 동시에 표현할 수 있다.
라. 보기 쉽고 이해하기 쉽다.

=> HIPO : 하향식

[기-99년10월][기-02년5월]
61. 데이터 모델링에 있어서 ERD(Entity Relationship Diagram)는 무엇을 나타내고자 하는가?
가. 데이터 흐름의 표현
나. 데이터 구조의 표현
다. 데이터 구조들과 그들간의 관계들을 표현
라. 데이터 사전을 표현

=> 개체관계도 : 데이터베이스에서 배웠죠.

[기-99년10월][기-02년3월]
62. 자료 흐름 중심의 설계 절차들을 올바른 순서로 나열한 것은?

1. 자료흐름도를 프로그램 구조로 사상한다.
2. 흐름의 경계를 표시한다.
3. 정보흐름의 유형을 설정한다.
4. 제어계층을 분해시켜서 정의한다.
5. 경험적 방법으로 구체화시킨다.

가. 1-2-3-4-5 나. 3-2-1-4-5
다. 4-5-3-2-1 라. 4-5-1-2-3

=> 아래 3가지 순서만이라도 기억하세요.
정보흐름의 유형 설정(데이터설계) -> 자료흐름도를 프로그램 구조로 사상(구조설계) -> 제어계층을 분해시켜서 정의(절차설계)

[기-99년8월][기-02년5월][기-07년5월]
63. 사용자 인터페이스 설계시 오류 메시지나 경고에 관한 다음의 지침 중 옳지 않은 것은 어느 것인가?
가. 메시지는 이해하기 쉬어야 한다.
나. 오류로부터 회복을 위한 구체적인 설명이 제공되어야 한다.
다. 오류로 인해 발생될 수 있는 부정적인 내용은 가급적 피한다.
라. 소리나 색 등을 이용하여 듣거나 보기 쉽게 의미 전달을 하도록 한다.

[기-00년3월][기-04년5월]
64. 설계품질을 평가하기 위해서는 반드시 좋은 설계에 대한 기준을 세워야 한다. 다음 중 좋은 기준이라고 할 수 없는 것은?
가. 설계는 모듈적이어야 한다.
나. 설계는 자료와 프로시저에 대한 분명하고 분리된 표현을 포함해야 한다.
다. 소프트웨어 요소들간의 효과적 제어를 위해 설계에서 계층적 조직이 제시되어야 한다.
라. 설계는 서브루틴이나 프로시저가 전체적이고 통합적이 될 수 있도록 유도되어야 한다.

=> 좋은 설계 : 모듈, 분리, 계층, 독립

[기-00년7월][기-02년9월][기-03년3월]
65. 소프트웨어 설계를 위한 지침에 대한 설명으로 거리가 먼 것은?
가. 소프트웨어 요소간의 효과적 제어를 위해 설계에서 계층적 자료조건이 제시되어야한다.
나. 설계는 종속적인 기능적 특성을 가진 모듈화로 유도 되어야 한다.
다. 소프트웨어는 논리적으로 특별한 기능과 부기능을 수행하는 요소들로 나누어져야 한다.
라. 설계는 자료와 프로시저에 대한 분명하고 분리될 표현을 포함해야 한다.

=> 종속이라는 설명이 틀렸습니다.

[기-00년10월][기-04년3월]
66. 소프트웨어 구조와 관련된 용어로, 주어진 한 모듈을 제어하는 상위 모듈 수를 나타내는 것은?
가. Modularity 나. Subordinate
다. Fan-in 라. Super-Ordinate

=> Fan-in : 상위, '인상'

[기-02년3월][기-05년3월][기-06년5월]
67. 구조적 프로그래밍에서 사용하는 기본적인 제어구조에 해당하지 않는 것은?
가. 순차(sequence) 나. 반복(iteration)
다. 호출(call) 라. 선택(selection)

[기-99년8월][기-07년3월][기-05년3월][기-01년6월]
68. 나씨-슈나이더만(Nassi-Schneiderman) 도표는 구조적 프로그램을 표현하기 위해 고안되었다. 이 방법에서 알고리즘의 제어구조는 3가지로 충분히 표현될 수 있는데 이에 속하지 않는 것은 어느 것인가?
가. 선택, 다중선택(if ~ then ~ else, case)
나. 반복 (repeat ~ until, while, for)
다. 분기(goto, return)
라. 연속(sequential)

[기-03년8월][기-05년5월]
69. NS(Nassi-Schneiderman) chart에 대한 설명으로 거리가 먼 것은?
가. 논리의 기술에 중점을 둔 도형식 표현 방법이다.
나. 연속, 선택 및 다중 선택, 반복 등의 제어논리 구조로 표현한다.
다. 주로 화살표를 사용하여 논리적인 제어구조로 흐름을 표현 한다.
라. 조건이 복합되어 있는 곳의 처리를 시각적으로 명확히 식별하는데 적합하다.

=> NS chart : 박스 다이어그램

[기-08년9월][기-08년3월][기-04년3월][기-03년3월][기-07년5월][기-07년3월][기-01년3월]
70. 좋은 모듈이 되기 위한 응집도와 결합도에 대한 설명으로 옳은 것은?
가. 모듈의 응집도와 결합도 모두가 높아야 한다.
나. 모듈의 응집도는 높아야 하고 결합도는 낮아야 한다.
다. 모듈의 응집도는 낮아야 하고 결합도는 높아야 한다.
라. 모듈의 응집도와 결합도 모두가 낮아야 한다.

=> 모듈은 독립적입니다.

[기-05년9월][기-99년8월]
71. 효과적인 모듈화 설계 방안이 아닌 것은?
가. 응집도를 높인다.
나. 결합도를 낮춘다.
다. 복잡도와 중복을 피한다.
라. 예측 불가능하도록 정의한다.

[기-02년3월][기-03년5월][기-04년9월][기-00년7월]
72. 결합도(coupling)가 강한 순서대로 옳게 나열된 것은?
가. 내용 결합도>공통 결합도>제어 결합도>스탬프 결합도>데이터 결합도
나. 공통 결합도>내용 결합도>제어 결합도>데이터 결합도>스탬프 결합도
다. 데이터 결합도>내용 결합도>제어 결합도>공통 결합도>스탬프 결합도
라. 공통 결합도>내용 결합도>제어 결합도>스탬프 결합도>데이터 결합도

=> 데<스<제<외<공<내
단순한 데이터에 의한 결합은 낮지만 내용이 결합되는 것은 높다.

[기-03년3월][기-00년3월]
73. 한 모듈이 다른 모듈의 내부 기능 및 그 내부 자료를 참조하는 경우, 이를 무슨 결합이라고 하는가?
가. 내용 결합 나. 제어 결합
다. 공통 결합 라. 스탬프 결합

=> 내용 결합도 : 다른 모듈의 내부기능,내부자료 참조

[기-05년3월][기-99년10월]
74. 모듈 결합도(Coupling)에 관한 설명으로 옳지 않은 것은?
가. 자료결합(Data Coupling) - 모듈간의 인터페이스가 자료요소로만 구성된 경우
나. 스탬프결합(Stamp Coupling) - 모듈간의 인터페이스로 배열이나 레코드 등의 자료구조가 전달된 경우
다. 내용결합(Content Coupling) - 한 모듈이 다른 모듈의 일부분을 참조 또는 수정하는 경우
라. 제어결합(Control Coupling) - 한 모듈이 다른 모듈에게 제어요소를 전달하고 여러 모듈이 공통 자료영역을 사용하는 경우

=> 라.번 보기의 제어결합의 설명 중 앞부분은 맞는 설명이고 뒤 부분의 설명은 공통결합의 설명이다.

[기-06년9월][기-03년8월][기-00년10월]
75. 한 모듈과 다른 모듈간 의 상호 의존도 또는 두 모듈 사이의 연관 관계를 의미하는 것은?
가. 신뢰도 나. 충실도 다. 응집도 라. 결합도

[기-01년3월][기-05년3월]
76. 두 모듈이 동일한 자료구조를 조회하는 경우의 결합성이며 자료구조의 어떠한 변화, 즉 포맷이나 구조의 변화는 그것을 조회하는 모든 모듈 및 변화되는 필드를 실제로 조회하지 않는 모듈에까지도 영향을 미치게 되는 결합성은?
가. data coupling 나. stamp coupling
다. control coupling 라. content coupling

=> 스탬프 결합도 : 자료구조

[기-01년6월][기-04년3월][기-02년3월][기-99년8월]
77. 응집력이 강한 것부터 약한 순서로 옳게 나열된 것은?
가. sequential → functional → procedural → coincidental → logical
나. procedural → coincidental → functional → sequential → logical
다. functional → sequential → proedural → logical → coincidental
라. logical → coincidental → functional → sequential → procedural

=> 우<논<시<절<교<순<기

[기-03년3월][기-06년5월]
78. 모듈을 이루고 있는 각 요소들이 공통의 목적을 달성하기 위하여 얼마나 관련이 있는가를 나타내는 것을 무엇이라고 하는가?
가. 결합도(coupling) 나. 응집도(cohesion)
다. 구조도(structure) 라. 일치도(unity)

=> 모듈 하나를 설명하고 있습니다.

[기-99년10월][기-05년5월]
79. 소프트웨어 개발을 위한 프로그래밍 언어의 선정기준으로 거리가 먼 것은?
가. 개발담당자의 경험과 지식 나. 대상업무의 성격
다. 과거의 개발실적 라. 4세대 언어 여부

=> 무조건 4세대 언어를 선택할 필요는 없죠.

[기-01년9월][기-02년3월][기-04년9월][기-07년5월]
80. 소프트웨어 개발 방법론에서 구현(Implementation)에 대한 설명으로 가장 적절한 것은?
가. 요구사항 분석 과정 중 모아진 요구사항을 옮기는 것
나. 시스템이 무슨 기능을 수행하는지에 대한 시스템의 목표기술
다. 프로그래밍 또는 코딩이라고 불리며 설계 명세서가 컴퓨터가 알 수 있는 모습으로 변환되는 과정
라. 시스템이나 소프트웨어 요구 사항을 정의하는 과정

=> 구현 : 프로그래밍

[기-00년7월][기-01년9월][기-06년3월]
81. 모듈안의 작동을 자세히 관찰할 수 있으며, 프로그램 원시 코드의 논리적인 구조를 커버(cover)하도록 테스트 케이스를 설계하는 프로그램 테스트 방법은?
가. 블랙 박스 테스트 나. 화이트 박스 테스트
다. 알파 테스트 라. 베타 테스트

=> 화이트 박스 테스트 : 구조 테스트(코드,논리적구조)

[기-00년7월][기-05년3월]
82. McCabe에 의해 제안된 소프트웨어의 복잡성 측정에 대한 설명으로 옳지 않은 것은?
가. 영역은 그래프의 평면에서 둘러 쌓여진 부분으로 묘사될 수 있다.
나. 영역의 수는 경계된 영역들과 그래프 외부의 비 경계지역의 수를 계산한다.

다. 모듈크기의 실제 상한선은 존재하지 않는다.
라. V(G)는 영역의 수를 결정함으로써 계산되어 진다.

=> 그래프의 크기는 둘러 쌓여진 영역의 수를 계산한다.
모듈의 크기는 모듈이 구성하고 있는 명령어의 라인수를 의미하는데 보통 100라인 이하가 적당하다.

[기-99년4월][기-04년5월]
83. 블랙박스 테스팅을 통해 발견하기 힘든 오류는?
가. 성능 오류 나. 부정확한 기능
다. 인터페이스 오류 라. 논리구조상의 오류

=> 화이트 박스 테스트 : 구조 테스트(코드,논리적구조)

[기-01년6월][기-03년5월]
84. 블랙 박스 검사에 해당하지 않는 것은?
가. 데이터 흐름 검사(data flow testing)
나. 동치 분할 검사(equivalence partitioning testing)
다. 원인 효과 그래픽 기법(cause effect graphic-technique)
라. 비교 검사(comparison testing)

=> 동치분할검사, 경계값 분석, 오류예측검사, 비교검사, 원인-효과 그래프 검사
* 까만 밤, 동경에 오면 비를 원없이 볼 수 있다.

[기-01년3월][기-05년3월]
85. 제품이 수행할 특정 기능을 알기 위해서 각 기능이 완전히 작동되는 것을 입증하는 검사로서, 기능 검사라고도 하는 것은?
가. 블랙 박스 검사 나. 그린 박스 검사
다. 블루 박스 검사 라. 화이트 박스 검사

=> 화이트박스 검사:구조, 블랙박스 검사:기능

[기-99년10월][기-04년3월]
86. 상향식 통합테스트(Bottom-up Integration Test)의 과정이 옳게 나열된 것은?

① 드라이버라는 제어프로그램의 작성
② 낮은 수준의 모듈들을 클러스터로 결합
③ 클러스터의 검사
④ 드라이버를 제거하고 클러스터를 상위로 결합

가. ①→②→③→④ 나. ②→①→③→④
다. ②→③→①→④ 라. ①→②→④→③

=> 절차 : 하위 모듈을 클러스터로 결합 -> 드라이버라는 제어 프로그램 작성 -> 클러스터 검사
-> 드라이버 제거하고 클러스터를 상위로 결합

[기-00년3월][기-02년5월]
87. 소프트웨어 시험의 목적은 오류를 찾아내는데 있다. 이의 종류로는 단위시험, 통합시험, 검증 시험, 그리고 시스템 시험이 있는데 이 중에서 소프트웨어가 요구사항에 맞는지를 추적해 보는데 중점을 두고 있는 시험방법은 무엇인가?
가. 단위 시험 나. 통합 시험
다. 검증 시험 라. 시스템 시험

=> 검증 검사 : 요구사항을 충족하는지 검사 (블랙 박스 테스트 기법 사용)

[기-00년10월][기-02년3월]
88. 소프트웨어 검사 단계를 올바른 순서로 나열한 것은?

㉠ 설계 검사	㉡ 요구사항 검사
㉢ 코드 검사	㉣ 시스템 검사

가. ㉠-㉡-㉢-㉣ 나. ㉢-㉠-㉡-㉣
다. ㉡-㉢-㉣-㉠ 라. ㉡-㉣-㉠-㉢

=> 검사 순서 : 단위(코드) -> 통합(설계) -> 검증(요구사항) -> 시스템

[기-00년10월][기-03년3월]
89. 검증 시험을 하는데 있어 알파 테스트란?
가. 사용자의 장소에서 개발자가 직접 시험을 한다.
나. 사용자의 장소에서 개발자와 사용자가 실 데이터를 가지고 공동으로 시험한다.
다. 개발자의 장소에서 개발자가 시험을 하고 사용자는 지켜본다.
라. 개발자의 장소에서 사용자가 시험을 하고 개발자는 뒤에서 결과를 지켜본다.

[기-02년5월][기-05년5월]
90. 검증시험(Validation Test)을 하는데 있어 Beta Test에 대한 설명으로 옳은 것은?
가. 사용 부서에서 개발 담당자가 시험한다.
나. 개발부서와 사용 부서가 공동으로 시험한다.
다. 개발 부서에서 개발자가 시험을 한다.
라. 실 업무를 가지고 사용자가 직접 시험한다.

[기-03년5월][기-05년5월]
91. 하향식 통합에 있어서 모듈간의 통합시험을 하기 위해 일시적으로 필요한 조건만을 가지고 임시로 제공는 시험용 모듈을 무엇이라고 하는가?
가. Driver 나. Stub
다. Sub-Program 라. Dummy-Program

=> Stub : 시험용 모듈

[기-07년5월][기-06년5월][기-01년9월][기-03년3월][기-03년8월][기-99년4월]
92. 소프트웨어 개발 단계에서 가장 많은 비용이 소요되는 단계는?
가. 계획 단계 나. 분석 단계
다. 구현 단계 라. 유지보수 단계

[기-05년9월][기-01년6월]
93. 소프트웨어 유지보수의 유형에 해당하지 않는 것은?
가. 수정보수(Corrective maintenance)
나. 기능보수(Functional maintenance)
다. 완전화보수(Perfective maintenance)
라. 예방보수(Preventive maintenance)

=> 완전화 보수, 기능 보수 (Perfective) : 기능 개선, 가장 큰 비중 차지(Win98 -> Win 2000 -> Win XP)

[기-06년3월][기-03년5월][기-00년7월]
94. 소프트웨어 유지보수에 관련된 설명으로 옳지 않은 것은?
가. 유지보수는 소프트웨어가 인수, 설치된 후 발생하는 모든 공학적 작업을 말한다.
나. 유지보수는 원인에 따라 수리(corrective)보수,

적응(adaptive)보수, 완전화(perfective)보수, 예방(preventive)보수 등이 있다.
다. 소프트웨어에 가해지는 연결을 제어 관리하는 것을 형상관리(configuration management)라고 한다.
라. 소프트웨어 비용 중 유지보수 비용은 개발비용 보다 적다.

[기-06년3월][기-00년7월]
95. 소프트웨어 유지보수 작업의 목적으로 부적절한 것은?
가. 하자보수 나. 환경적응
다. 예방조치 라. 설계수정

=> 설계 수정은 소프트웨어 구조를 바꾸는 것이므로 유지보수 목적이라 할 수 없다.

[기-07년3월][기-05년3월]
96. 소프트웨어의 문서(document) 표준이 되었을 때, 개발자가 얻는 이득으로 가장 거리가 먼 것은?
가. 시스템 개발을 위한 분석과 설계가 용이하다.
나. 프로그램 유지보수가 용이하다.
다. 프로그램의 확장성이 있다.
라. 프로그램 개발 인력이 감소된다.

= 문서화는 개발 과정을 기록으로 남기는 것이다. 문서화를 하기 때문에 개발 인력이 감소되는 것은 아니죠.

[기-03년3월][기-01년9월]
97. 외계인 코드(Alien Code)를 방지하기 위한 방법으로 가장 적합한 것은?
가. 프로그램 내에 문서화(Documentation)를 철저하게 해두어야 한다.
나. 자료흐름도(DFD)를 상세히 그려야 한다.
다. 프로그램 완성시 testing을 확실하게 해야 한다.
라. 프로그램시 반드시 visual tool을 사용해야 한다

=> 외계인 코드:아주 오래되어(15년 이상) 유지보수 작업이 어려운 프로그램 (방지법 : 문서화)

[기-02년3월][기-04년3월][기-99년8월]
98. 소프트웨어 라이프사이클 단계중 가장 오랜 시간이 걸리며, 대부분의 비용을 차지하는 단계는?
가. 타당성 검토 단계 나. 운용 및 유지 보수 단계
다. 기본설계 단계 라. 실행 단계

[기-02년9월][기-01년3월]
99. 소프트웨어 유지보수에 대한 설명으로 옳지 않은 것은?
가. 소프트웨어 유지보수 비용은 개발비용보다 일반적으로 적다.
나. 소프트웨어 유지보수를 용이하게 하려면 시험용이성,이해성,수정용이성, 이식성이 고려되어야 한다.
다. 소프트웨어 유지보수의 과정은 유지보수요구,현시스템에 대한 이해, 수정 및 시험순을 반복하여 일어난다.
라. 소프트웨어 유지보수는 기능개선 ,하자보수. 환경적응, 예방조치를 목적으로 소프트웨어의 수명을 연장시키는 작업이다.

=> 유지보수비용 > 개발비용

[기-08년5월][기-00년7월][기-04년3월][기-06년5월][기-06년9월][기-07년5월]
100. 객체 지향 개념 중 하나 이상의 유사한 객체들을 묶어 공통된 특성을 표현한 데이터 추상화를 의미하는 것은?
가. 메소드(method) 나. 클래스(class)
다. 상속성(inheritance) 라. 추상화(abstraction)

[기-05년3월][기-99년8월]
101. 객체 지향 개념에서 오퍼레이션(operation)은 무엇을 변화시키는가?
가. 어트리뷰트(attribute) 나. 클래스 (class)
다. 오브젝트(object) 라. 메시지(message)

=> 행위는 상태를 변화시킨다.
객체 (Object)
① 데이터 : 객체가 가지고 있는 상태 (속성, Attribute, 변수, 자료구조)
② 연산자 : 객체의 데이터를 처리하는 행위 (메소드, Method, 동작, Operation, 함수, 프로시져)

[기-00년10월][기-03년3월]
102. 객체지향 기법에서 메소드는 어느 시점에 시작되어 지는가?
가. 사용자 명령어가 입력될때
나. OS에 의하여 인터럽트가 감지될 때
다. 특별한 데이터 값을 만날 때
라. 오브젝트로부터 메시지를 받을 때

=> 메소드(method)
객체가 메시지를 받아 실행해야 할 객체의 구체적인 연산

[기-01년3월][기-02년9월]
103. 객체지향 기법에서 메시지(message)의 전달은 어떻게 이루어지는가?
가. 어트리뷰트(attribute)에서 어트리뷰트로
나. 오브젝트(object)에서 어트리뷰트로
다. 오브젝트(object)에서 오브젝트로
라. 클래스(class)에서 데이터(data)로

=> 메시지 (Message)
객체들 간에 상호작용을 하는데 사용되는 수단

[기-01년6월][기-03년3월][기-04년9월][기-07년9월]
104. 객체지향 시스템에서 전통적 시스템의 함수(function) 또는 프로시저(procedure)에 해당하는 연산기능을 무엇이라고 하는가?
가. 메소드(method) 나. 메시지(message)
다. 모듈(module) 라. 패키지(package)

[기-08년5월][기-08년3월][기-07년3월][기-06년3월]
105. 객체지향 개념에서 객체가 메시지를 받아 실행해야 할 객체의 구체적인 연산을 정의한 것은?
가. 클래스 나. 메시지 다. 인스턴스 라. 메소드

[기-01년9월][기-00년3월][기-04년3월][기-07년5월]
106. 객체지향 시스템에서 자료부분과 연산(또는 함수) 부분 등 정보처리에 필요한 기능을 한 테두리로 묶는 것을 무엇이라고 하는가?
가. 정보은닉 나. 클래스 다. 캡슐화 라. 통합

[기-02년9월][기-06년5월]

107. 객체를 이용하여 데이터와 연산들을 하나의 단위로 묶는 기법은?
가. instance 나. polymorphism
다. inheritance 라. encapsulation

=> 캡슐화 (Encapsulation)

[기-08년3월][기-04년9월][기-06년3월]
108. 객체지향 개념에서 연관된 데이터와 함수를 함께 묶어 외부와 경계를 만들고 필요한 인터페이스만을 밖으로 드러내는 과정을 무엇이라고 하는가?
가. 메시지 나. 캡슐화 다. 상속 라. 다형성

=> 캡슐화 : 묶어

[기-07년9월][기-07년3월]
109. 객체지향에서 캡슐화에 대한 설명으로 잘못된 것은?
가. 재사용이 용이하다.
나. 인터페이스를 단순화 시킬 수 있다.
다. 응집도가 향상된다.
라. 결합도가 높아진다.

=> 캡슐화로 응집도는 높아지고, 결합도는 낮아진다.

[기-99년4월][기-02년9월][기-05년5월]
110. 객체지향 설계에 있어서 정보은폐(information hiding)의 근본적인 목적은?
가. 코드를 개선하기 위하여
나. 프로그램의 길이를 짧게 하기 위하여
다. 고려되지 않은 영향(side effect)들을 최소화하기 위하여
라. 인터페이스를 최소화하기 위하여

[기-08년5월][기-00년10월][기-02년3월]
111. 객체는 다른 객체로부터 자신의 자료를 숨기고 자신의 연산만을 통하여 접근을 허용하는 것은 무엇이라 하는가?
가. abstraction 나. information hiding
다. modularity 라. typing

=> 정보은폐 (Information Hiding) : 객체는 다른 객체로부터 자신의 자료를 숨기고 자신의 연산만을 통하여 접근을 허용하는 것

[기-99년10월][기-04년3월]
112. 객체지향 기법에서 상속(INHERITANCE)의 결과로서 얻을 수 있는 가장 주요한 이점은?
가. 모듈 라이브러리의 재이용
나. 객체지향 DB를 사용할 있는 능력
다. 클래스와 오브젝트들을 재사용할 수 있는 능력
라. 프로젝트들을 보다 효과적으로 관리할 수 있는 능력

=> 상속 (Inheritance) : 상위 클래스의 메소드와 속성을 하위 클래스가 물려받는 것

[기-08년9월][기-05년3월]
113. 객체 지향 소프트웨어 공학의 상속성에 대해 바르게 설명한 것은?
가. 상위 클래스의 메소드와 속성을 하위 클래스가 물려받는 것을 말한다.
나. 데이터와 데이터를 조작하는 연산을 하나로 묶는 것을 말한다.
다. 객체 클래스로부터 만들어진 하나의 인스턴스이다.
라. 변수가 취할 수 있는 여러 가지 특성 중의 하나를 결정받는 것을 말한다.

[기-01년6월][기-04년3월]
114. 객체지향 소프트웨어 개발모형의 개발 단계로 옳은 것은?

㉠ 설계 ㉡ 구현 ㉢ 계획 ㉣ 분석 ㉤ 테스트 및 검증

가. ㉢→㉠→㉣→㉡→㉤ 나. ㉢→㉣→㉡→㉠→㉤
다. ㉢→㉡→㉣→㉠→㉤ 라. ㉢→㉣→㉠→㉡→㉤

[기-01년9월][기-00년7월]
115. 객체지향 분석에 대한 설명으로 옳지 않은 것은?
가. 분석가에게 주요한 모델링 구성요소인 클래스, 객체, 속성, 연산들을 표현해서 문제를 모형화시킬 수 있게 해준다.
나. 객체지향 관점은 모형화 표기법의 전후관계에서 객체의 분류, 속성들의 상속, 그리고 메시지의 통신 등을 결합한 것이다.
다. 객체는 클래스로부터 인스턴스화 되고, 이 클래스를 식별하는 것이 객체지향분석의 주요한 목적이다.
라. E-R 다이어그램은 객체지향분석의 표기법으로는 적합하지 않다.

[기-05년5월][기-05년3월][기-06년5월][기-01년3월][기-02년3월][기-07년3월][기-06년9월]
116. 람보우(Rumbaugh)의 객체지향 분석절차를 바르게 나열한 것은?
가. 객체 모형 → 동적 모형 → 기능 모형
나. 객체 모형 → 기능 모형 → 동적 모형
다. 기능 모형 → 동적 모형 → 객체 모형
라. 기능 모형 → 객체 모형 → 동적 모형

[기-06년3월][기-04년9월][기-00년7월][기-02년5월][기-07년5월]
117. 객체 모형(object method), 동적 모형(dynamic model), 기능 모형(functional model)의 3개 모형으로 구성되어 있는 개체지향 분석 기법은?
가. Rambaugh method 나. Wirls-Rrock method
다. Jacobson method 라. Coad & Yourdon method

[기-08년5월][기-01년6월][기-03년8월][기-05년3월][기-07년9월]
118. 람바우의 객체 지향 분석 모델링(modeling)에 해당하지 않는 것은?
가. relational modeling 나. object modeling
다. functional modeling 라. dynamic modeling

[기-04년5월][기-01년3월]
119. 객체지향 설계에 대한 설명으로 옳지 않은 것은?
가. 객체지향 설계에 있어 가장 중요한 문제는 시스템을 구성하는 객체와 속성, 연산을 인식하는 것이다.
나. 시스템 기술서의 동사는 객체를, 명사는 연산이나 객체 서비스를 나타낸다.
다. 객체지향 설계를 문서화할 때 객체와 그들의 부객체(sub-object)의 계층적 구조를 보여주는 계층차트를 그리면 유용하다.
라. 객체는 순차적으로(Sequentially) 또는 동시적으로

(Concurrently) 구현될 수 있다.

=> 동사(연산), 명사(객체)

[기-08년3월][기-08년9월]
120. OMT(Object Modeling Technique)에서 다수 프로세스간의 데이터 흐름을 중심으로 처리 과정을 자료 흐름도로 나타내는 것과 관계되는 것은?
가. Dynamic Modeling 나. Function Modeling
다. Object Modeling 라. Class Modeling

=> OMT(object modeling technique,객체모델링기법) 방법론은 Rumbaugh 등이 제안
기능 모델링 : 자료흐름도를 이용하여 동작 기술

[기-08년3월][기-99년4월]
121. 다음 중 소프트웨어를 재사용함으로써 얻는 이점이 아닌 것은?
가. 개발시간과 비용을 단축시킨다.
나. 소프트웨어 개발의 생산성을 높인다.
다. 프로젝트 실패의 위험을 줄여 준다.
라. 새로운 개발 방법론의 도입이 쉽다.

=> 이미 만들어진 프로그램을 사용하므로 다른 개발 방법론을 도입하기 어렵다.

[기-07년3월][기-03년8월]
122. 소프트웨어 재사용의 이점으로 볼 수 없는 것은?
가. 개발 비용을 감소시킨다.
나. 프로그램 언어가 종속적이다.
다. 소프트웨어 품질을 향상시킨다.
라. 프로그램 생성 지식을 공유하게 된다.

=> 다양한 프로그램 언어 중에서 하나를 선택해서 프로그램을 개발합니다. 개발된 프로그램을 재사용할 경우 동일한 프로그램 언어를 사용해야만 생산성이 높습니다. 한 마디로 재사용할 경우에는 프로그램 언어를 선택할 수 없으므로 프로그램에 종속적입니다. 이것은 재상용의 문제점입니다.

[기-06년9월][기-05년3월][기-02년9월][기-99년10월]
123. 소프트웨어의 재사용(reusability)에 대한 효과와 거리가 먼 것은?
가. 사용자의 책임과 권한부여
나. 소프트웨어 품질 향상
다. 생산성 향상
라. 구축 방법에 대한 지식의 공유

[기-06년5월][기-04년3월]
124. 소프트웨어의 재사용으로 얻어지는 이익이 아닌 것은?
가. 표준화의 원칙을 무시할 수 있다.
나. 프로젝트의 개발 위험을 줄여줄 수 있다.
다. 프로젝트의 개발기간과 비용을 줄일 수 있다.
라. 개발자의 생산성을 향상시킬 수 있다.

[기-08년5월][기-03년5월]
125. 소프트웨어 컴포넌트(Component) 재사용의 이점이라고 볼 수 없는 항목은?
가. 소프트웨어의 품질 향상
나. 개발 담당자의 생산성 향상
다. 개발 비용의 절감
라. 응용소프트웨어의 보안 유지

=> 컴포넌트(부품) 재사용은 소프트웨어 재사용과 같은 의미입니다.

[기-03년3월][기-07년9월]
126. 소프트웨어 재사용에 대한 설명으로 옳지 않은 것은?
가. 개발 시간과 비용을 감소시킨다.
나. 프로젝트 실패의 위험을 줄여 준다.
다. 재사용 부품의 크기가 작을수록 재사용 확률이 낮다.
라. 소프트웨어 개발자의 생산성을 증가시킨다.

=> 예를 들어 [사칙연산 계산기]를 하나의 부품으로 재사용하기 보다는 [덧셈],[나눗셈],[뺄셈],[곱셈]으로 크기를 작게 하면 여러 프로그램에서 필요한 연산 부품만 이용할 수 있으므로 재사용 확률이 높아진다.

[기-07년5월][기-03년5월]
127. 다음 설명에 해당하는 것은?

> - 기존 시스템을 이용하여 보다 나은 시스템을 구축하고 새로운 기능을 추가하여 소프트웨어 성능을 향상시킴
> - 주요 활동으로 분석, 개조, 역공학, 이식 등이 있음

가. 소프트웨어 재공학(Software Re-engineering)
나. 소프트웨어 분석(Software analysis)
다. 소프트웨어 프로그래밍(Software Programming)
라. 소프트웨어 개발(Software Development)

[기-01년9월][기-05년5월][기-03년8월]
128. 소프트웨어의 위기를 해결하기 위해 개발의 생산성이 아닌 유지보수의 생산성으로 해결하려는 방법을 의미하는 것은?
가. 소프트웨어 재사용
나. 소프트웨어 재공학
다. 클라이언트/서버 소프트웨어 공학
라. 전통적 소프트웨어공학

[기-03년3월][기-00년10월]
129. 소프트웨어 리엔지니어링(reengineering)의 목표 중 거리가 먼 것은?
가. 복잡한 시스템을 다루는 방법 구현
나. 다른 뷰의 생성
다. 기존 시스템의 해킹
라. 잃어버린 정보의 복구 및 제거

=> 소프트웨어 재공학 (Re-engineering)
소프트웨어의 위기를 해결하기 위해 개발의 생산성이 아닌 유지보수의 생산성으로 해결하려는 방법

[기-08년3월][기-99년4월]
130. 소프트웨어 재공학의 필요성이 대두된 주된 이유는?
가. 요구사항 분석의 문제 나. 설계의 문제
다. 구현의 문제 라. 유지보수의 문제

[기-08년3월][기-06년3월][기-02년3월][기-00년10월]
131. 현재 프로그램으로부터 데이터, 아키텍처, 그리고 절차에 관한 분석 및 설계 정보를 추출하는 과정은?
가. 재공학(re-engineering)

나. 역공학(reverse engineering)
다. 순공학(forward engineering)
라. 재사용(reuse)

=> 역공학 : 현재 프로그램에서 분석/설계 정보 추출

[기-08년5월][기-02년9월]
132. 소프트웨어 분석하여 소프트웨어 개발과정과 데이터 처리과정을 설명하는 분석 및 설계 정보를 재발견하거나 다시 만들어 내는 작업을 무엇이라 하는가?
가. 순공학 나. 역공학 다. 재구축 라. 전공학

[기-03년3월][기-00년3월]
133. 클라이언트/서버(Client/Server) 모델에서의 소프트웨어 개발에 대한 설명으로 옳지 않은 것은?
가. 사용자의 요구사항은 서버의 데이터베이스 시스템에 영향을 미친다.
나. 병목현상을 없애기 위해 비즈니스 로직을 분리하여 관리할 수 있다.
다. 미들웨어의 사용은 서버와 클라이언트의 작업량을 증가시켰다.
라. 대부분 네트워크로 연결되어 있고 인증 작업을 필요로 한다.

=> 클라이언트/서버(Client/Server) 모델은 다수의 서버용 컴퓨터 시스템을 네트워크로 연결한 구조이다. 이는 각 클라이언트(고객 및 사용자)에서 요구하는 작업을 부담을 줄이고 신뢰성 및 확장성을 고려한 경제적인 분산처리 시스템이다. 이런 시스템을 효과적으로 사용하기 위해서는 특성에 맞는 적절한 소프트웨어가 사용자에게 제공되어야 하는데 이러한 소프트웨어들을 총칭해서 미들웨어라고 한다.

[기-05년3월][기-01년6월]
134. CASE에 대한 설명으로 옳지 않은 것은?
가. 소프트웨어의 개발과정을 자동화함으로써 생산성을 증대시키고자 하는 목적으로 개발되었다.
나. CASE는 소프트웨어 개발의 모든 단계에 걸쳐 일관된 방법론을 지원한다.
다. CASE를 사용함으로 개발의 표준화를 지향하고, 자동화의 이점을 얻을 수 있다.
라. CASE는 시스템의 개발 속도를 빠르게 하지만 재사용성은 떨어진다.

=> 모듈의 재사용성이 향상

[기-08년3월][기-03년8월][기-06년5월][기-00년3월]
135. CASE(Computer-Aided Software Engineering)에 대한 설명으로 옳지 않은 것은?
가. 소프트웨어 개발의 작업들을 자동화하는 것이다.
나. 소프트웨어 도구와 방법론의 결합이다.
다. 소프트웨어의 생산성 문제를 해결할 수 있다.
라. 개발과정이 빠른 대신 재사용성이 떨어진다.

[기-07년5월][기-06년3월][기-05년3월][기-02년5월]
136. 소프트웨어 생명 주기의 전체 단계를 연결시켜 주고 자동화 시켜 주는 통합된 도구를 제공해 주는 것은?
가. UIMS 나. CASE 다. OOD 라. SADT

[기-05년9월][기-99년4월]
137. CASE(Computer Aided Software Engineering)에 대한 설명 중 틀린 것은?
가. CASE는 상위(upper) CASE, 중위(midium) CASE, 하위(Lower) CASE, 통합(integrated) CASE의 4가지 형태로 나눌 수 있다.
나. 통합 CASE는 소프트웨어 개발 주기 전체과정을 지원한다.
다. 상위 CASE는 요구분석과 설계단계를 지원한다.
라. 하위 CASE는 코드를 작성하고 테스트하며 문서화하는 과정을 지원한다.

=> CASE 분류 : 상위,하위,통합

[기-04년5월][기-02년3월][기-00년7월]
138. 소프트웨어 개발 과정에서 사용되는 요구 분석, 설계, 구현, 검사 및 디버깅 과정을 컴퓨터와 전용의 소프트웨어 도구를 사용하여 자동화하는 것을 무엇이라고 하는가?
가. CAT(Computer Aided Testing)
나. CAD/CAM(Computer Aided Design and Manufacturing)
다. CASE(Computer Aided Software Engineering)
라. CAI(Computer Aided Instruction)

[기-08년9월][기-02년5월]
139. CASE 도구의 정보저장소(Repository)에 대한 설명으로 거리가 먼 것은?
가. 일반적으로 정보저장소는 도구들과 생명 주기 활동, 사용자들, 응용 소프트웨어들 사이의 통신과 소프트웨어 시스템 정보의 공유를 향상시킨다.
나. 초기의 소프트웨어 개발 환경에서는 사람이 정보 저장소 역할을 했지만 오늘날에는 응용프로그램이 정보 저장소 역할을 담당한다.
다. 정보 저장소는 도구들의 통합, 소프트웨어 시스템의 표준화, 소프트웨어 시스템 정보의 공유, 시스템웨어 재사용성의 기본이 된다.
라. 소프트웨어 시스템 구성 요소들과 시스템 정보가 정보 저장소에 의해 관리되므로 소프트웨어 시스템의 유지보수가 용이해진다.

=> CASE의 정보 저장소 : 개발 과정 동안에 모아진 정보를 보관하여 관리하는 곳 -> 유지보수 용이
* 소프트웨어 개발에 관련된 정보저장소의 역할은 운영체제가 담당한다.