<c언어>

1. Printf("%s", &b[6]);
    1. 0~5인덱스 무시. 6~끝 까지의 문자열 출력
2. 랜덤값
    1. rand()%6 +1
    2. 1~7사이 난수 발생. 즉, 0~6 사이 인덱스 맞출려면 h[n-1] 해야 맞음.
3. C++
    1. count << "b" -> b출력
    2. delete a -> ~a() 실행

<자바>

1. 상속관계 호출
    1. 부모에서 print()호출해도 자식의print() 가 호출됨
2. 16진수 계산
    1. %x = 16진수 표기법 (10진수를 16진수로 변환)
    2. 52 % 16 = 3 ... 4(나머지)
    3. 3(몫) % 16 = 0 ... 3(나머지)
    4. 따라서 51 의 16진수 -> 34
3. 랜덤값
    1. Math.random()*10); //0 ~ 10 사이
    2. Math.random()은 0.0~ 1.0 사이난수 발생
4. 배열
    1. 행이0~2, 열이 0~4 가지는 구조라면, `a[3][5]` 가 필요

<파이썬>

1. 리스트 연산
    1. append(10) -> 리스트 맨 끝에 10 추가
    2. remove(2) -> 리스트 값중 숫자2 삭제
    3. a[2] -> 인덱스 2번의 값 선택 (리스트는 0인덱스부터 시작)
2. count(값) -> 값의 요소 수를 반환
3. index(10) -> 10의 위치를 반환.
4. copy() -> 깊은복사해 각각의 메모리 주소를 가지게 한다.
5. pop(위치) -> 위치에 있는 값 출력 후 요소 삭제
6. set의 pop() -> 세트의 값 출력인데, 순서 모름
7. 반복문
    1. range(시작 숫자, 끝 숫자) -> 끝 숫자 포함 안됨
    2. 즉, range(1,11) 이면 1~10까지 숫자 반복임.
    3. c = [a[i] + b[i] for i in range(4)] -> 0~3 까지 숫자 반복해 c의 배열 [0,1,2] 번째에 들어감
8. 합계
    1. sum(a) : 배열 또는 리스트 전체의 합 구하는 함수
9. 리스트 더하기
    1. append() : 맨뒤에 더하기
    2. insert(index,값) : 원하는 위치에 값넣기
10. +: 리스트에 값 추가

<sql>

1. 연산함수
    1. 집합함수 count(), sum(),avg() 등은null 갯수 포함 안함
2. 레코드 수
    1. select count(*) 한 레코드의 수는 1이다.
    2. count(*) 과 4 값만나오니까 레코드는 1이다.
3. 필드명
    1. 무조건 as 쓰기.
    2. a.자격증명 -> a.자격증명 as 자격증명
4. IN( ) 사용
    1. select * from A where 학번 IN(3,4)
    2. 3,4 학년만 출력
5. 중복제거
    1. select distinct 학년 from 학생
6. 인덱스 스키마 생성
    1. create index 인덱스명 on 테이블명(속성명)
7. 순위
    1. select rank(점수) : 2위,2위,2위,5위 -> 중복이면 그만큼 순위 밀림
    2. select row_number(점수) : 1위,2위,3위 -> 중복관계없이
    3. select DENSE_RANK() OVER (ORDER BY 점수 DESC) as 점수 : 2위,2위, 3위 -> 중복인 경우 같은 순위

<DDL>

1. 데이터 구조를 정의하는데 사용
2. Create
    1. create 스키마 on 테이블명(속성)
    2. create view 사원뷰 as (select ...)
3. Drop
    1. drop [스키마, table] 테이블명
4. Alter table
    1. Alter table 테이블명 ADD 컬럼명
    2. alter table 테이블명 modify 컬럼명 NUMBER(6)
    3. alter table 테이블명 modify A INTEGER PRIMARY KEY

<DCL> - Data Control Language

1. 데이터의 보안, 무결성, 회복, 병행 수행 제어 등을 정하는데 사용하는 언어
2. Commit
    1. commit
3. Rollback
    1. rollback to 세이브포인트
4. Revoke
    1. revoke 등급 on 테이블명 from 사용자
5. Grant
    1. Grant 사용자등급 on 테이블명 to 사용자

2. Grant ALL ON A TO 사용자 WITH GRANT OPTION -> 다른 사람에게 권한 줄수있도록 권한부여

<DML> - Data Manipulation Language

1. 사용자가 실질적으로 저장된 데이터를 처리 할때 사용
2. select * from A
3. Insert into A values (a,b....)
4. Delete from A where..
5. Update A set 주소 = a

<TCL> 트랜잭션 컨트롤 언어

1. Commit : 트랜잭션을 메모리에 영구적 저장
2. Rollback : 트랜잭션 내역을 무효화
3. Checkpoint : 롤백위한 시점

<합집합>

1. Union : 합집합. 두 select문 조회결과 통합해 모두 출력 (중복행 하나만 출력)
2. Union all : 두 select문 조회결과 통합해 모두 출력 (중복행 그대로 출력)
3. Insersect : 두 select문 조회결과 중 공통행만 출력
4. Except : 첫select문 조회결과에서 두번째select문 결과 제외한것만 출력

<Join>

1. Left Outer : 왼쪽 모든데이터, 오른쪽 일치 데이터만 출력(왼쪽 출력시 오른쪽 필드 값 없는 경우 Null처리)
2. Right Outer : 오른쪽 모든 데이터, 왼쪽 일치 데이터만 출력
3. Full Outer : 양쪽 모든 데이터 출력

<문자열 찾기 like >

1. % : 0개 이상 문자열과 일치
2. ▢ : 1개의 문자와 일치
3. [^] : 1개의 문자와 불일치
4. _ : 특정위치 1개 문자와 일치

<관계 대수>

- 순수 관계 연산자
- 셀렉트, 파이, 조인, 디비전
    1. 셀렉트(시그마) : 선택 조건 만족하는 튜플을 새 릴레이션으로 만드는 연산 .가로행 가져오기.
    2. 프로젝트(파이) : 제시된 속성값만 추출하여 새 릴레이션으로 만드는 연산. 중복 발생시 제거. 세로행 가져오기
    3. 조인( ) : 두 릴레이션 합쳐 새 릴레이션 생성. 중복 허용
    4. 디비전(÷) : R과 S 릴레이션이 있을때 R의 속성이 S의 속성값을 가진 튜플에서 S가 가진 속성 제외한 속성을 구하는 연산.
- 일반 관계 연산자
    - 카디션곱, insertion, 유니온, difference

- 카디션곱 : 교차곱으로 차수(열)는 더하고 카디널리티(행)은 곱한다. 문자 X사용.

<연산자 순위>

1. 증 산 시 관 비 논 조 대 순
2. 감 술 프트 계 트 리 건 입 서

<자료사전 기호>

1. = : 정의
2. ∘ : 연결
3. ( ) : 생략
4. { } : 반복
5. ☐ : 선택
6. ** : 주석

<UML>

정적 다이어그램	
인스턴스를 특정 시점의 객체와 객체 사이의 관계 표현	객체
클래스가 복합구조 가질때 내부 구조 표현	복합
컴포넌트 사이 종속성 표현과 물리적 요소 위치 표현	배치
의존관계 나타냄	컴포넌트
클래스 모델 요소들 그룹화	패키지

동적 다이어그램	
사용자 측면에서 요구분석해 기능 모델링 작업에 사용	유스케이스
상호작용하는 시스템이나 객체들이 주고받는 메시지 표현	시퀀스
객체들간 주고받는 메시지와 객체간의 관계까지 표현	커뮤니케이션
객체가 자신이 속한 클래스의 상태변화에 따라 어떻게 변하는지 표현	상태
객체 처리 로직이나 조건에 따른 처리 흐름 순서에 따라 표현	활동
객체 상태 변화와 시간 제약 명시적으로 표현	타이밍
상호작용 다이어그램 간의 제어 표현	상호작용

uml 관계 종류	
2개 이상의 사물이 서로 관련되어 있음을 표현	연관관계
하나의 사물이 다른 사물에 포함	집합관계
집합관계의 특수한 형태로, 포함하는 사물의 변화가 포함되는 사물에 영향	포함관계
하나의 사물이 다른 사물에 비해 일반적인지 구체적인지 표현	일반화 관계
사물 사이에 연관은 있으나 필요에 의해서 서로에게 영향주는 짧은 시점에만 관계성립	의존관계

uml 관계 종류

사물이 할 수 있거나 해야하는 기능으로 서로를 그룹화	실체화 관계

<공통 모듈 테스트>

<화이트박스> - 응용프로그램 내부구조와 동작을 검사하는 소프트웨어 테스트

화이트 박스 테스트

테스트 케이스 설계자가 논리적 복잡성을 측정할 수 있게 함	기초경로 검사
논리적인 조건 초점	조건 검사
반복구조 초점	루프 검사
변수정의나 위치 초점	데이터 흐름

<화이트박스 검증 기준>

소스 코드 모든 부분을 한번이상 수행	문장 (statement)
조건식에 상관없이 개별 조건식이 참 / 거짓 인 경우 한번 이상 수행되도록 구성하는 검증	조건 (condition)
모든 조건문에 대해 조건식이 참/거짓인 경우가 한 번 이상 수행 되도록 구성하는 검증	분기
모두 만족하는 조건검증으로, 조건문이 true false인 경우에 따라 조건 검증 기준의 입력 데이터를 구분하는 검증 기법	분기 / 조건

<블랙박스>

과거 경험이나 감각 이용	오류-예측
정상/비정상 동작의 예상 결과를 동일한 수 만큼 테스트	동치분할 Fquivalence partitioning
입력값의 경계에 있는 값으로 테스트	경계값 분석 (Boundary value)
다른 버전의 프로그램에 동일한 값을 넣어 동일한 결과가 나오는지 테스트	비교 (comparison)
분석 후 최적의 데이터로 테스트	원인-효과

<테스트 목적에 따른 분류>

실패 유도후 정상적 복귀 여부	회복 테스트(Recovery Testing)
보안적 결함을 미리 점검	안전 테스트(Security Testing)
응답하는 시간, 업무량, 반응속도 측정	성능 테스트(Performance Testing)
내부 논리 경로, 소스 코드의 복잡도를 평가	구조 테스트(Structure Testing)
오류를 제거하거나 수정한곳에 새로운 오류 확인	회귀 테스트(Regression Testing)
변경된 시스템과 동일한 데이터를 입력후 결과 비교	병행 테스트(Parallel Testing)

<단위 모듈 구현 원리>

변경 가능성 있는 모듈을 다른 모듈로부터 은폐	정보은닉
복잡한 문제를 분해하고 모듈 단위로 문제해결	분할과 정복
자료구조 엑세스하고 함수 내에 자료구조 표현 내역 은폐	데이터 추상화
낮은결합도와 높은 응집도	모듈 독립성

<단위 모듈 재사용성>

기존 기능 개선 또는 재활용	재공학
sw에 대한 디버깅 같은 분석통해 알고리즘을 역으로 분석해 재구성	역공학
기존 시스템 참조하여 새로운 시스템 개발	재개발

<GOF 디자인 패턴 >

생성

구체적 클래스에 의존하지 않고 인터페이스 통해 연관있는 객체들의 그룹으로 생서해 추상적으로 표현	추상
인스턴스를 조합하여 객체를 생성	빌더
객체 생성을 서브 클래스에서 구현하도록 분리하여 캡슐화	팩토리
비용이 큰 경우 사용하는 패턴으로, 원본 객체를 복사하는 방법으로 객체 생성	프로토 타입
클래스 내에서 인스턴스가 하나뿐임을 보장하는 패턴, 하나 객체 생성하면 어디서든 해당 객체 참조가능	싱글톤

구조

다른 클래스가 사용할 수 있도록 연결 돕는 패턴	어댑터
기능과 구현을 두개의 별도 클래스로 구현해 서로 독립적으로 확장할 수 있도록 함	브릿지
복합객체와 단일객체를 구분없이 다루고자 할 때 사용	컴포지트
객체간의 결합을 통해 능동적으로 기능들을 확장	데코레이터
상위 인터페이스에 기능 구현하고 하위에서 쉽게 사용	파싸드
메모리 절약을 위해 인스턴스 가능한 공유해서 사용	플라이웨이드
객체와 객체 사이에서 연결 돕기위한 역할 수행	프록시

행위

요청처리 못하면 다음 객체가 처리	책임연쇄
각종 명령들을 분리해 단순화	커맨드
언어문법 표현 정리	인터프리터

행위	
접근 잦은 객체는 동일 인터페이스 사용	반복자
객체 사이 상호작용을 캡슐화	중재자
특정시점에서 내부 상태를 객체화해 객체를 특정 시점으로 돌리는 기능 제공	메멘토
객체의 상태가 변화하면 객체에 상속되어있는 곳에 상태를 전달하는 패턴	옵서버
알고리즘 개별 생성해 원할때마다 변경하는 패턴	전략
상위에선 골격, 하위에선 처리하는 패턴	템플릿메소드
처리기능 분리해 별도로 구성	방문자

<응집도 및 결합도>

<응집도> -(Cohesion)	
서로 관련 없는 요소로 구성	우연적 (Concidental)
유사 성격이나 특정 형태로 된 요소로 구성	논리적 (Logical)
특정 시간에 처리되야 하는 요소로 구성	시간적 (Temporal)
다수의 관련 기능을 가질 때 그 기능을 순차적으로 수행	절차적 (Procedual)
동일한 입출력으로 다른 기능 수행	교환적 (Communication)
내부 결과물을 다음 활동의 입력값으로 사용	순차적 (Sequental)
모든 기능이 단일 목적 위해 수행	기능적 (Function)

<결합도>	
다른 모듈의 기능 및 자료를 직접 사용	내용
공유 데이터 영역을 여러 모듈이 사용	공통
어떤 모듈의 변수를 외부 모듈이 사용	외부
다른 모듈 내부의 논리 흐름을 제어하기 위한 제어 요소 전달 (권리전도현상 발생)	제어
모듈간 인터페이스 배열, 레코드의 자료구조 전달	스탬프
모듈간 인터페이스가 자료 요소만으로 구성	자료

<트랜잭션 특징>

<트랜잭션 특징>	
트랜잭션의 연산은 DB에 모두 반영되도록 완료(Commit) 되거나 반영되지 않도록 복구(Rollback) 되어야 한다.	원자성 (Atomicity)
트랜잭션이 실행을 성공적으로 완료하면 언제나 일관성있는 데이터베이스 상태로 변환함	일관성 (consistency)

<트랜잭션 특징>

트랜잭션 병행 실행되는 경우, 하나의 트랜잭션 실행중에 다른 트랜잭션이 연산에 끼어들 수 없음.	독립성 (Isolation)
성공적으로 완료된 트랜잭션은 시스템이 고장나더라도 영구적으로 반영	영속성 (Durability)

<DB 보안 3요소>

<DB 보안 3요소>

시스템의 자원과 정보는 인가된 사용자에게만 접근이 허락된다.	기밀성 (Confidentialit)
시스템 자원은 오직 인가된 사용자만 사용하며 수정 및 변경할 수 있다.	무결성 (Integrity)
인가받은 사람은 언제 어디서든 사용할 수 있다.	가용성 (Availability)
사용자의 신원을 인정	인증(authentication)
인증된 주체에게 접근권한 허용	인가(Authorization)
데이터 송수신 증거 제시하는 방법	부인방지

<소스코드 도구>

정적 분석 도구 (pmd, Cppcheck 빼고 나머지가 크로스 플랫폼 지원)

미사용 변수, 최적화 안된 코드 ,결함 유발 코드 검사 (리눅스 , 윈도우)	Pmd
C와 C++ 코드에 대한 메모리 누수, 오버플로우 검사 (윈도우)	Cppcheck
중복코드 , 복잡도, 코딩설계등을 분석하는 소스 분석 통합 플랫폼	SonarQube
자바 코드에 대해 코드 표준 검사	Checkstyle
다양한 언어의 코드 복잡도 분석	Ccm
자바 언어의 소스 코드 복잡도 분석	Cobertura

동적 분석 도구

Valgrind 프래임워크 및 STP 기반으로 구현	Avalanche
프로그램 내에 존재하는 메모리 및 쓰레드 결함 분석	Valgrind

< 빌드도구>

< 빌드도구>

Ant
Make
Maven

< 빌드도구>

groovy 기반으로 한 안드로이트 앱 개발환경에서 사용되는 도구. 명령어 일들을 모아 태스크 단위로 처리	Gradle
java기반 서블릿 컨테이너에서 실행되는 도구. 분산빌드나 테스트 기능 지원	Jenkins

<인터페이스 구현 검증 도구>

<인터페이스 구현 검증 도구>

Java, C++ 등 다양한 언어 지원하는 단위테스트 프레임워크	XUnit
컴포넌트 재사용 등 다양한 환경 제공 프래임워크	STAF
웹 기반 테스트 프래임워크	FitNesse
FitNesse의 협업기능과 STAF를 통합한 NHN프레임워크	NTAF
다양한 브라우저 지원하는 웹 어플리케이션 테스트 프레임워크	Selenium
Ruby사용	Watir
Db에이전트를 통해 애플리케이션 모니터링	스카우터
애플리케이션 운영,안정화 전까지 모든 생명주기 성능 모니터링	제니퍼

<페이지 교체 알고리즘>

<페이지 교체 알고리즘>

가장 오랫동안 사용하지 않을 데이터 삭제	OPT
먼저 들어온 것부터 삭제	FIFO
가장 오랫동안 사용되지 않은거부터 삭제	LRU
참조 횟수가 가장 적은거부터 삭제	LFU
참조 횟수가 가장 많은것부터 삭제	MFU
최근 사용하지 않은 것부터 삭제	NUR