컴퓨터일반

문 1. 인터프리터(Interpreter) 방식의 언어로 옳지 않은 것은? 2

① JavaScript
② C
③ Basic
④ LISP

[해설]

- 컴파일 언어 : FORTRAN, COBOL, ALGOL, PL/1, PASCAL, C, ADA 등
- 인터프리터 언어 : BASIC, APL, SNOBOL, LISP, JavaScript 등

문 2. CPU 스케줄링 기법 중 라운드 로빈(Round Robin) 방식에 대한 설명으로 옳지 않은 것은? 4

① 선점 스케줄링 기법이다.
② 여러 프로세스에 일정한 시간을 할당한다.
③ 시간할당량이 작으면 문맥 교환수와 오버헤드가 증가한다.
④ FIFO(First-In-First-Out) 방식 대비 높은 처리량을 제공한다.

[해설]

- RR(Round Robin, 라운드 로빈)
① FCFS를 선점형 스케줄링으로 변형한 기법이다.
② 대화형 시스템에서 사용되며, 빠른 응답시간을 보장한다.
③ RR은 각 프로세스가 CPU를 공평하게 사용할 수 있다는 장점이 있지만, 시간할당량의 크기는 시스템의 성능을 결정하므로 세심한 주의가 필요하다.

문 3. 프로세서의 수를 늘려도 속도를 개선하는 데 한계가 있다는 주장으로서, 병렬처리 프로세서의 성능 향상의 한계를 지적한 법칙은? 2

① 무어의 법칙(Moore's Law)
② 암달의 법칙(Amdahl's Law)
③ 구스타프슨의 법칙(Gustafson's Law)
④ 폰노이만 아키텍처(von Neumann Architecture)

[해설]

- 암달의 법칙(Amdahl's Law) : 병렬처리 프로그램에서 차례로 수행되어야하는 비교적 적은 수의 명령문들이, 프로세서의 수를 추가하더라도 그 프로그램의 실행을 더 빠르게 할 수 없도록 속도향상을 제한하는 요소를 갖고 있다는 것이다.

문 4. 교착상태 발생의 조건에 대한 설명으로 옳지 않은 것은? 4

① 상호 배제 조건: 최소한 하나의 자원이 비공유 모드로 점유되며, 비공유 모드에서는 한 번에 한 프로세스만 해당 자원을 사용할 수 있다.
② 점유와 대기 조건: 프로세스는 최소한 하나의 자원을 점유한 채, 현재 다른 프로세스에 의해 점유된 자원을 추가로 얻기 위해 반드시 대기해야 한다.
③ 비선점 조건: 프로세스에 할당된 자원은 사용이 끝날 때까지 다른 프로세스가 강제로 빼앗을 수 없다.
④ 순환 대기 조건: 대기 체인 내 프로세스들의 집합에서 이전 프로세스는 다음 프로세스가 점유한 자원을 대기하고, 마지막 프로세스는 자원을 대기하지 않아야 한다.

[해설]

- 환형대기(순환대기,Circular Wait)
① 공유자원들을 여러 프로세스에게 순서적으로 분배한다면, 시간은 오래 걸리지만 교착상태는 발생하지 않는다. 그러나 프로세스들에게 우선순위를 부여하여 공유자원 할당의 사용시기와 순서를 융통성 있게 조절한다면, 공유자원의 점유와 대기는 환형대기 상태가 될 수 있다.
② 여러 프로세스들이 공유자원을 사용하기 위해 원형으로 대기하는 구성으로, 앞이나 뒤에 있는 프로세스의 자원을 요구한다.

문 5. CPU(중앙처리장치)의 성능 향상을 위해 한 명령어 사이클 동안 여러 개의 명령어를 동시에 처리할 수 있도록 설계한 CPU구조는? 1

① 슈퍼스칼라(Superscalar)
② 분기 예측(Branch Prediction)
③ VLIW(Very Long Instruction Word)
④ SIMD(Single Instruction Multiple Data)

[해설]

- 슈퍼스칼라 (superscalar) : CPU 내에 파이프라인을 여러 개 두어 명령어를 동시에 실행하는 기술이다. 파이프라인과 병렬 처리의 장점을 모은 것으로, 여러 개의 파이프라인에서 명령들이병렬로 처리되도록 한 아키텍처이다. 여러 명령어들이 대기 상태를 거치지 않고 동시에 실행될 수 있으므로 처리속도가 빠르다.

문 6. 캐시기억장치 접근시간이 20 ns, 주기억장치 접근시간이 150 ns, 캐시기억장치 적중률이 80 %인 경우에 평균 기억장치 접근시간은? (단, 기억장치는 캐시와 주기억장치로만 구성된다) 2

① 32 ns
② 46 ns
③ 124 ns
④ 170 ns

[해설]

- 캐시기반 평균 기억장치 접근 시간

Taverage = Hhit-ratio × Tcache + (1 - Hhit-ratio) × Tmain

- Taverage : 평균 기억장치 접근 시간
- Hhit-ratio : 적중률
- Tcache : 캐시 기억장치 접근 시간
- Tmain : 주기억장치 접근 시간

Taverage = 0.8 * 20 ns + (1 - 0.8) * 150ns = 46 ns

문 7. 아날로그 컴퓨터에 대한 설명으로 옳지 않은 것은? 1

① 입력형식은 부호, 코드화된 숫자, 문자, 기호이다.
② 출력형식은 곡선, 그래프 등이다.
③ 미적분 연산방식을 가지며, 정보처리속도가 빠르다.
④ 증폭회로 등으로 회로 구성을 한다.

[해설]

- 디지털 컴퓨터와 아날로그 컴퓨터의 비교

항목	디지털 컴퓨터	아날로그 컴퓨터
입력 형태	숫자, 문자	전류, 전압, 온도
출력 형태	숫자, 문자	곡선, 그래프
연산 형식	산술·논리 연산	미·적분 연산
구성 회로	논리회로	증폭 회로
프로그래밍	필요	불필요
정밀도	필요한 한도까지	제한적임
기억 기능	있음	없음
적용분야	범용	특수 목적용

문 8. RAID(Redundant Array of Inexpensive Disks)에 대한 설명으로 옳지 않은 것은? 3

① RAID-0은 디스크 스트라이핑(Disk Striping) 방식으로 중복 저장과 오류 검출 및 교정이 없는 방식이다.
② RAID-1은 디스크 미러링(Disk Mirroring) 방식으로 높은 신뢰도를 갖는다.
③ RAID-4는 데이터를 비트(bit) 단위로 여러 디스크에 분할하여 저장하는 방식이며, 별도의 패리티(parity) 디스크를 사용한다.
④ RAID-5는 별도의 패리티 디스크 대신 모든 디스크에 패리티 정보를 나누어 기록하는 방식이다.

[해설]

- RAID 4 (Parity) : 한 드라이브에 패리티 정보를 저장하고 나머지 드라이브에 데이터를 블록단위로 분산한다. 패리티 정보는 어느 한 드라이브가 장애가 발생했을 때 데이터 복구가 가능하다. 데이터를 읽을 때는 RAID 0에 필적하는 우수한 성능을 보이나, 저장할 때는 매번 패리티 정보를 갱신하기 때문에 추가적인 시간이 필요하다.

문 9. 다음 재귀 함수를 동일한 기능의 반복 함수로 바꿀 때, ㉠과 ㉡에 들어갈 내용을 바르게 연결한 것은? 3

```
int func (int n) {                    //재귀 함수
    if (n == 0)
        return 1;
    else
        return n * func (n - 1);
}
int iter_func (int n) {               //반복 함수
    int f = 1;
    while ( ㉠ )
        ㉡
    return f;
}
```

	㉠	㉡
①	n < 0	f = f * n--;
②	n < 0	f = f * n++;
③	n > 0	f = f * n--;
④	n > 0	f = f * n++;

[해설]

- 코드에서 재귀함수에 대한 부분에서 if (n == 0)이 종료조건이 되고, 조건이 만족하지 않을때 수행되는 return n * func (n - 1);을 보면 n값에서 1을 감소하고 되부름을 하고있으므로 factorial 알고리즘에 해당된다.
- 반복함수에서는 n > 0 이 참일 때 f = f * n--; 문장을 반복하여 처리한다면 factorial 알고리즘을 수행할 수 있다.

문 10. 데이터의 종류 및 처리에 대한 설명으로 옳지 않은 것은? 1

① 크롤링(Crawling)을 통해 얻은 웹문서의 텍스트 데이터는 대표적인 정형 데이터(Structured Data)이다.
② XML로 작성된 IoT 센서 데이터는 반정형 데이터(Semi-structured Data)로 분류할 수 있다.
③ 반정형 데이터는 데이터 구조에 대한 메타 데이터(Meta-data)를 포함한다.
④ NoSQL과 Hadoop은 대규모 비정형 데이터(Unstructured Data) 처리에 적합하다.

[해설]

- 크롤링(Crawling)은 웹 페이지를 그대로 가져와서 거기서 데이터를 추출해 내는 행위라 할 수 있다. 즉, 특정 웹 사이트에서 원하는 정보를 자동으로 수집하는 것이며, 크롤링을 통해 얻은 웹문서의 텍스트 데이터는 대표적인 반정형 데이터(Semi-structured Data)라 할 수 있다.

문 11. 페이지 부재율(Page Fault Ratio)과 스래싱(Trashing)에 대한 설명으로 옳은 것은? 4

① 페이지 부재율이 크면 스래싱이 적게 일어난다.
② 페이지 부재율과 스래싱은 전혀 관계가 없다.
③ 스래싱이 많이 발생하면 페이지 부재율이 감소한다.
④ 다중 프로그램의 정도가 높을수록 스래싱이 증가한다.

[해설]

- 스래싱(Thrashing) : 페이지부재가 지나치게 발생하여 프로세스가 수행되는 시간보다 페이지 이동에 시간이 더 많아지는 현상이다. 다중프로그래밍 정도를 높이면, 어느 정도까지는 CPU 이용률이 증가되지만, 스래싱에 의해 CPU 이용률은 급격히 감소된다.
- 페이지 부재율이 크면 스래싱이 많이 일어난다.

문 12. 전자상거래 관련 기술 중 고객의 요구에 맞춰 자재조달에서부터 생산, 판매, 유통에 이르기까지 공급사슬 전체의 기능통합과 최적화를 지향하는 정보시스템은? 3

① ERP(Enterprise Resource Planning)
② EDI(Electronic Data Interchange)
③ SCM(Supply Chain Management)
④ KMS(Knowledge Management System)

[해설]

- ERP(Enterprise Resource Planning) : 기업 내 생산, 물류, 재무, 회계, 영업과 구매, 재고 등 경영 활동 프로세스들을 통합적으로 연계해 관리해 주며, 기업에서 발생하는 정보들을 서로 공유하고 새로운 정보의 생성과 빠른 의사결정을 도와주는 전사적자원관리시스템 또는 전사적통합시스템을 말한다.
- EDI(Electronic Data Interchange) : 기업간에 데이터를 효율적으로 교환하기 위해 지정한 데이터와 문서의 표준화 시스템이다.
- SCM(Supply Chain Management) : 기업에서 원재료의 생산·유통 등 모든 공급망 단계를 최적화해 수요자가 원하는 제품을 원하는 시간과 장소에 제공하는 공급망 관리를 말한다.
- KMS(Knowledge Management System) : 조직 내 전문화 또는 관리적 활동을 대상으로 정보나 데이터가 아닌 조직의 지식을 창출, 수집, 조직화하고 공유하기 위해 구축되는 일련의 시스템으로 정의된다.

문 13. 프로토콜과 이에 대응하는 TCP/IP 프로토콜 계층 사이의 연결이 옳지 않은 것은? 2

① HTTP – 응용 계층
② SMTP – 데이터링크 계층
③ IP – 네트워크 계층
④ UDP – 전송 계층

[해설]

- Application layer(응용 계층) : 사용자들이 응용 프로그램을 사용할 수 있도록 다양한 서비스를 제공한다. 인터넷 브라우저를 이용하기 위한 HTTP 서비스, 파일 전송 프로그램을 위한 FTP 서비스, 메일 전송을 위한 SMTP, 네트워크 관리를 위한 SNMP 등의 서비스를 제공한다.

문 14. 관계 데이터베이스 스키마 STUDENT(SNO, NAME, AGE)에 대하여 다음과 같은 SQL 질의 문장을 사용한다고 할 때, 이 SQL 문장과 동일한 의미의 관계대수식은? (단, STUDENT 스키마에서 밑줄 친 속성은 기본키 속성을, 관계대수식에서 사용하는 관계대수 연산자 기호 π는 프로젝트 연산자를, σ는 셀렉트 연산자를 나타낸다) 2

<SQL 질의문>

```
SELECT SNO, NAME
FROM STUDENT
WHERE AGE > 20;
```

① $\sigma_{SNO,NAME}(\pi_{AGE>20}(STUDENT))$
② $\pi_{SNO,NAME}(\sigma_{AGE>20}(STUDENT))$
③ $\sigma_{AGE>20}(\pi_{SNO,NAME}(STUDENT))$
④ $\pi_{AGE>20}(\sigma_{SNO,NAME}(STUDENT))$

[해설]

- 셀렉트(SELECT, σ) : 선택 조건을 만족하는 릴레이션의 수평적 부분집합(horizontal subset), 행의 집합

표기 형식 → σ<선택조건>(테이블 이름)

- 프로젝트(PROJECT, π) : 수직적 부분 집합(vertical subset), 열(column)의 집합

표기 형식 → π<속성 리스트>(테이블 이름)

문 15. 두 프로토콜 개체 사이에서 흐름제어와 오류제어 및 메시지 전달 등의 기능을 수행하며, 연결성과 비연결성의 두 가지 운용 모드를 제공하는 OSI 참조 모델 계층은? 3

① 데이터링크 계층(Datalink Layer)
② 네트워크 계층(Network Layer)
③ 전송 계층(Transport Layer)
④ 응용 계층(Application Layer)

[해설]

- Transport layer(전송 계층) : 전송층은 수신측에 전달되는 데이터에 오류가 없고 데이터의 순서가 수신측에 그대로 보존되도록 보장하는 연결 서비스의 역할을 하는 종단간(end-to-end) 서비스 계층이다. 종단간의 데이터 전송에서 무결성을 제공하는 계층으로 응용 계층에서 생성된 긴 메시지가 여러 개의 패킷으로 나누어지고, 각 패킷은 오류없이 순서에 맞게 중복되거나 유실되는 일 없이 전송되도록 하는데 이러한 전송 계층에는 TCP, UDP 프로토콜 서비스가 있다.

문 16. 소프트웨어 개발 언어에 대한 설명으로 옳지 않은 것은? 3

① C#은 마이크로소프트 닷넷 프레임워크를 지원하는 객체지향 언어이다.

② Python은 인터프리터 방식의 객체지향 언어로서 실행시점에 데이터 타입을 결정하는 동적 타이핑 기능을 갖는다.

③ Kotlin은 그래픽 요소를 강화한 게임 개발 전용 언어이다.

④ Java는 컴파일된 프로그램이 JVM상에서 인터프리터 방식으로 실행되는 플랫폼 독립적 프로그래밍 언어이다.

[해설]

- Kotlin : 안드로이드 스튜디오 개발사인 Jet Brains에서 2011년에 공개한 언어이다. 코틀린은 JVM에서 구동되는 언어로 자바와 상호운용할 수 있도록 만들으며, 자바를 완전히 대체할 수 있는 언어가 되는 것이 코틀린의 주목적이라 할 수 있다. 구글이 안드로이드 공식 언어로 코틀린을 추가했다.

문 17. 소프트웨어 시스템은 기능 관점, 동적 관점 및 정보 관점으로 분류할 수 있다. 동적 관점에서 시스템을 기술할 때 사용할 수 있는 도구로 옳지 않은 것은? 2

① 사건 추적도(Event Trace Diagram)

② 자료 흐름도(Data Flow Diagram)

③ 상태 변화도(State Transition Diagram)

④ 페트리넷(Petri Net)

[해설]

1. 기능관점(Function Space)
- 기능 모델은 시스템이 어떠한 기능을 수행하는가의 관점에서 시스템을 기술한다.
- 주어진 입력에 대하여 어떤 결과가 나오는가를 보여주는 관점이여 연산과 제약조건을 묘사한다.
- 기능 모델의 일반적인 표현 방법은 자료흐름도에 의하여 도식적으로 나타난다.

2. 동적 관점(Dynamic Space)
- 시간의 변화에 따른 시스템의 동작과 제어에 초점을 맞추어 시스템의 상태와 상태를 변하게 하는 원인을 묘사하는 것이다.
- 상태변화도(STD), 사건추적도(ETD), 페트리네트 : 동적관점을 기술할 때, 외부와의 상호작용이 많은 실시간 시스템들은 동적 관점에서 시스템이 기술되어야 할 때 쓰이는 도구이다.

3. 정보 관점(Information Space)
- 시스템에 필요한 정보를 보여줌으로써 시스템의 정적인 정보구조를 포착하는데 사용한다.
- 정보 모델은 특히 시스템의 데이터베이스를 분석하는 데 많이 사용되며 ER 모델이 대표적인 도구이다.

문 18. 다음에서 설명하는 네트워크 데이터 오류 검출 방법은? 3

> 송신측: 첫 번째 비트가 1로 시작하는 임의의 n+1비트의 제수를 결정한다. 그리고 전송하고자 하는 데이터 끝에 n비트의 0을 추가한 후 제수로 모듈로-2 연산을 한다. 그러면 n비트의 나머지가 구해지는데 이 나머지가 중복 정보가 된다.
>
> 수신측: 계산된 중복 정보를 데이터와 함께 전송하면 수신측에서는 전송받은 정보를 동일한 n+1제수로 모듈로-2 연산을 한다. 나머지가 0이면 오류가 없는 것으로 판단하고, 나머지가 0이 아니면 오류로 간주한다.

① 수직 중복 검사(Vertical Redundancy Check)

② 세로 중복 검사(Longitudinal Redundancy Check)

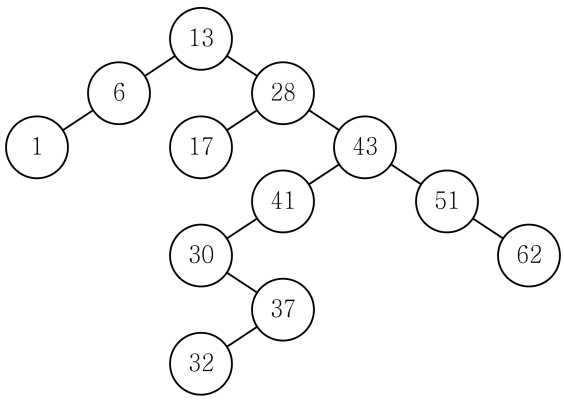
③ 순환 중복 검사(Cyclic Redundancy Check)

④ 체크섬(Checksum)

[해설]

- 순환 중복 검사(Cyclic Redundancy Check)
㉠ 전체 블록을 검사하며, 이진 나눗셈을 기반으로 한다.
㉡ 계산 방법
ⓐ 메시지는 하나의 긴 2진수로 간주하여, 특정한 이진 소수에 의해 나누어진다.
ⓑ 나머지는 송신되는 프레임에 첨부되며, 나머지를 BCC(Block Check Character)라고도 한다.
ⓒ 프레임이 수신되면 수신기는 같은 제수(generator)를 사용하여 나눗셈의 나머지를 검사하며, 나머지가 0이 아니면 에러가 발생했음을 의미한다.

문 19. 다음 이진검색트리에서 28을 삭제한 후, 28의 오른쪽 서브트리에 있는 가장 작은 원소로 28을 대치하여 만들어지는 이진검색트리에서 41의 왼쪽 자식 노드는? 4



① 13 ② 17

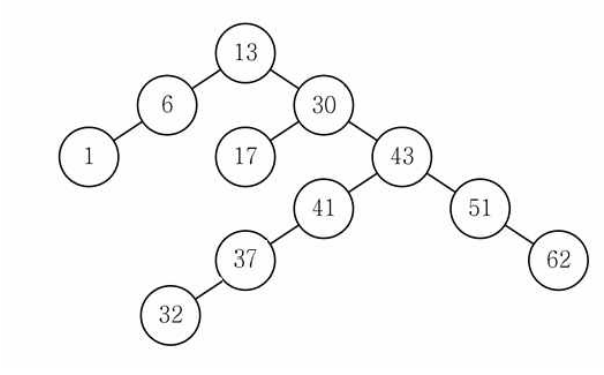
③ 32 ④ 37

[해설]

- 문제의 이진검색트리에서 28을 삭제한 후, 28의 오른쪽 서브트리에 있는 가장 작은 원소로 28을 대치하면 30으로 대치될 수 있다.
- 30으로 대치되면 원래 30의 자리에 37이 대치되고 37의 왼쪽자식으로

32가 들어간다.

- 노드 41의 왼쪽자식은 37이 되며, 최종 이진탐색트리는 아래와 같다.



문 20. 다음은 리눅스 환경에서 fork() 시스템 호출을 이용하여 자식 프로세스를 생성하는 C 프로그램이다. 출력 결과로 옳은 것은? (단, "pid = fork();" 문장의 수행 결과 자식 프로세스의 생성을 성공하였다고 가정한다) 1

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/types.h>
#include<errno.h>
#include<sys/wait.h>

int main(void) {
    int i=0, v=1, n=5;
    pid_t pid;

    pid = fork();

    if( pid < 0 ) {
        for(i=0; i<n; i++) v+=(i+1);
        printf("c = %d ", v);
    } else if( pid == 0 ) {
        for(i=0; i<n; i++) v*=(i+1);
        printf("b = %d ", v);
    } else {
        wait(NULL);
        for(i=0; i<n; i++) v+=1;
        printf("a = %d ", v);
    }
    return 0;
}
```

① b = 120, a = 6　　② c = 16, b = 120

③ b = 120, c = 16　　④ a = 6, c = 16

[해설]

- PID : Process IDentifier의 약자로, 운영 체제에서 프로세스를 식별하기 위해 할당하는 고유한 번호이다.
- fork() 함수 : 현재 실행되고 있는 프로세스를 복사해주는 함수이다.
- 헤더는 unistd.h 이고, 원형은 pid_t fork(void)이다.
- fork() 함수를 호출하고 성공하면 PID 값만 다른 똑같은 프로세스가 생성된다. 실행이 실패하면 -1이 반환되고, 성공하면 부모 프로세스에게는 자식 프로세스의 PID를, 자식 프로세스에는 0을 반환한다. 원본 프로세스가 부모 프로세스이고 복사된 프로세스가 자식 프로세스가 된다.

```
if( pid < 0 ) {                    // 에러 발생시 실행되는 부분
        for(i=0; i<n; i++) v+=(i+1);
        printf("c = %d ", v);
    } else if( pid == 0 ) {        // 자식 프로세스가 실행되는 부분
        for(i=0; i<n; i++) v*=(i+1);
        printf("b = %d ", v);
    } else {                       // 부모 프로세스가 실행되는 부분
        wait(NULL);                // 부모 프로세스는 자식 프로세스
        for(i=0; i<n; i++) v+=1;      가 종료될때까지 기다린다.
        printf("a = %d ", v);
    }
```

- if문 실행시 else if문이 실행되고, else문이 실행된다.