2017년 국가직 하반기 생활안전분야 9급 컴퓨터일반 풀이 by 호이호이꿀떡

정답 체크

01	02	03	04	05	06	07	08	09	10
②	④	④	②	①	④	④	③	②	③
11	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
④	①	④	①	③	②	③	②	④	②

문 1. TCP(Transmission Control Protocol)와 IP(Internet Protocol)에 대한 설명으로 옳지 않은 것은?

① TCP 는 호스트 사이에 신뢰성 있는 스트림(stream) 전송 서비스를 제공한다.
② IP 는 수신 측 IP 주소를 바탕으로 라우팅 테이블을 갱신한다.
③ TCP는 연결 지향형 프로토콜로서 실제 데이터를 전송하기 전에 연결을 설정한다.
④ IP는 신뢰성을 보장하지 않는 비연결 지향형 프로토콜이다.

답 ②

② 라우팅 테이블은 IP 주소와 IP 주소로 향하는 최적의 경로(라우팅 정보)를 담고 있는 테이블이다.
라우팅 테이블에도 목적지(수신 측) IP 주소가 담겨 있기는 하지만, 라우팅 테이블 정보를 갱신할 때 중요한 것은 라우팅 정보(경로)의 변경 유무이다. IP 주소가 변하지 않더라도 서로 이웃하는 라우터들끼리 정보를 주고 받으며, 라우팅 정보에 변경사항이 존재할 경우 라우팅 테이블을 갱신하기 때문이다.
따라서 선택지에서처럼 단순하게 IP 주소를 바탕으로 라우팅 테이블을 갱신한다는 말은 너무 포괄적이고 부족한 설명이라 옳은 설명이라고 볼 수 없다.

<오답 체크> ①③ TCP는 신뢰할 수 있고, 연결 지향형이다. 연결 지향(Connection-oriented)이란 호스트가 데이터를 교환하기 이전에 연결이 반드시 이루어져야 함을 의미한다.

④ IP는 비연결 지향형이다. 비연결 지향이라는 것은 데이터를 교환하기 이전에 세션이 확립되지 않음을 의미한다.

문 2. IPv4 주소를 클래스별로 분류했을 때, C 클래스에 해당하는 것은?

① 12.34.56.78
② 111.11.11.11
③ 123.12.31.12
④ 222.22.22.22

답 ④

A 클래스 0******* 으로 시작(0~127)
B 클래스 10****** 으로 시작(128~191)
C 클래스 110***** 으로 시작(192~223)
D 클래스 1110**** 으로 시작(224~239) 멀티캐스트용
E 클래스 1111**** 으로 시작(240~255) 미래를 위해 예비용

④ 222.22.22.22
222는 이진수로 11011110이므로 C 클래스

<오답 체크> ① 12.34.56.78
12는 이진수로 00001100이므로 A 클래스

② 111.11.11.11
111은 이진수로 01101111이므로 A 클래스

③ 123.12.31.12
123은 이진수로 01111011이므로 A 클래스

문 3. 운영체제의 스케줄링 기법에 대한 설명으로 옳지 않은 것은?

① FCFS(First-Come-First-Served) 스케줄링은 비선점(nonpreemptive) 방식으로 실행 중인 프로세스가 종료하면 준비 큐에서 가장 오래 대기한 프로세스를 다음 실행 프로세스로 선정한다.

② RR(Round-Robin) 스케줄링은 선점(preemptive) 방식으로 프로세스를 정해진 시간 할당량만큼 실행 후 종료하지 못하면 준비 큐로 이동시킨다.

③ 비선점 SJF(Shortest-Job-First) 스케줄링은 준비 큐에서 예상 전체 실행시간이 가장 짧은 프로세스를 다음 실행 프로세스로 선정한다.

④ 선점 SJF 스케줄링은 SRTF(Shortest-Remaining-Time-First) 스케줄링이라고 불리며 비선점 SJF 스케줄링에서 발생할 수 있는 기아상태(starvation) 문제를 해결한다.

답 ④

④ 선점 SJF 또는 SRT 스케줄링은 항상 실행시간이 짧은 프로세스가 우선하기 때문에, 실행시간이 긴 프로세스는 기아상태에 빠지게 된다. 비선점 SJF 방식도 마찬가지이다.
기아상태 문제를 해결하기 위해서는 에이징 기법을 적용한 HRN 스케줄링이나 RR(라운드 로빈) 방식을 사용해야 한다.

◈ 비선점(Non-Preemptive) 스케줄링 종류

▷ FCFS(First-Come First-Service) = FIFO(First In First Out)
준비상태 큐에 도착한 순서에 따라 차례로 CPU를 할당

▷ SJF(Shortest Job First)
실행시간이 가장 짧은 프로세스에 먼저 CPU를 할당하는 기법

▷ HRN(Hightest Responseratio Next)
우선순위 계산 공식 = 대기시간 + 서비스시간 / 서비스시간

▷ 기한부(Deadline)
프로세스에게 일정한 시간을 주어 그 시간 안에 프로세스를 완료하도록 하는 기법

▷ 우선순위(Priority)
준비상태 큐에서 기다리는 각 프로세스마다 우선순위를 부여하는 기법

▷ 에이징(Aging) 기법
기아 상태를 해결하기 위해, 기다린 시간에 비례하여 일정 시간이 지나면 우선순위를 한 단계씩 높여 가까운 시간 안에 자원을 할당받도록 하는 기법

◈ 선점(Preemptive) 스케줄링 종류

▷ 선점 우선순위
준비 상태 큐의 프로세스들 중에서 우선순위가 가장 높은 프로세스에게 먼저 CPU를 할당하는 기법

▷ SRT(Shortest Remaining Time)
비선점 기법 SJF 알고리즘을 선점 형태로 변경한 기법
현재 실행 중 프로세스의 남은 시간과 준비상태 큐에 새로 도착한 프로세스의 실행시간을 비교해 가장 짧은 실행 시간을 요구하는 프로세스에게 CPU를 할당하는 기법

▷ RR(Round Robin)
각 프로세스는 할당된 시간(Time Slice Quantum)동안만 실행한 후 실행이 완료되지 않으면 다음 프로세스에게 CPU를 넘겨주고 준비상태 큐의 가장 뒤로 배치함

▷ 다단계 큐(Multi level Queue)
프로세스를 특정 그룹으로 분류할 수 있는 경우 그룹에 따라 각기 다른 준비상태 큐를 사용하는 기법

▷ 다단계 피드백 큐(Multi level Feedback Queue)
특정 그룹의 준비상태 큐에 들어간 프로세스가 다른 준비상태 큐로 이동할 수 없는 다단계 큐 기법을 준비상태 큐 사이를 이동할 수 있도록 개선한 기법

문 4. 가상 객체와 실세계를 접목하여 현실감 있는 정보를 제공하는 기술은?

① 지리정보 시스템(geographical information system)
② 증강현실(augmented reality)
③ 생체인식(biometrics)
④ 사물인터넷(Internet of Things)

답 ②

② **증강현실**(AR, Augmented?Reality)
가상현실(VR, Virtual Reality)에서 파생된 기술로, 실제 환경에 가상 사물이나 부가정보를 합성하여 원래의 환경에 존재하는 사물처럼 보이도록 하는 컴퓨터 그래픽 기법

<오답 체크> ① **지리 정보 체계**(GIS, Geographic Information System)는 지리공간적으로 참조가능한 모든 형태의 정보를 효과적으로 수집, 저장, 갱신, 조정, 분석, 표현하여 의사결정에 반영할 수 있는 시스템이다.

③ **생체인식**(biometrics)
하나 이상의 고유한 신체적, 행동적 형질에 기반하여 사람을 인식하는 방식을 두루 가리킨다. 생체 인증 기술에 쓰이는 신체적 특성으로는 지문, 홍채, 얼굴, 정맥 등이 있으며 행동적 특성으로는 목소리, 서명 등이 있다.

④ **사물인터넷**(Internet of Things)
정보 통신 기술을 기반으로 실세계와 가상 세계의 다양한 사물들을 연결하여 진보된 서비스를 제공하기 위한 지능형 인프라 및 서비스 기술이다.

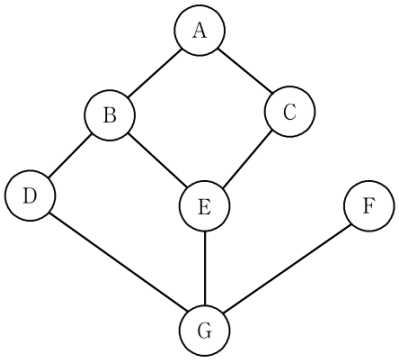
문 5. 10 진수 -11 을 5 비트 2 진수로 표현한 것은? (단, 부호 있는(signed) 2 진수는 2 의 보수로 표현된다)

① 10101
② 11101
③ 01101
④ 10100

답 ①

11을 다섯 자리 2진수를 바꾸면 01011 이 된다.
각 비트를 반전시키면 1의 보수 음수값이 되므로,
1의 보수 -11은 '10100' 이 된다.
1의 보수 음수값에 1을 더하면 2의 보수 음수값이 되므로,
2의 보수 -11은 '10101' 이 된다.

문 6. 다음 그래프의 정점 A 에서부터 깊이 우선 탐색(DFS: Depth First Search)과 너비 우선 탐색(BFS: Breadth First Search)을 수행할 때, 방문 순서를 옳게 짝지은 것은? (단, 방문하지 않은 인접 정점이 2 개 이상인 경우 알파벳 오름차순으로 방문한다)



① DFS : A-B-D-G-F-C-E
BFS : A-B-C-D-E-F-G
② DFS : A-B-D-G-F-C-E
BFS : A-B-C-D-E-G-F
③ DFS : A-B-D-G-E-C-F
BFS : A-B-C-D-E-F-G
④ DFS : A-B-D-G-E-C-F
BFS : A-B-C-D-E-G-F

답 ④

◈ **깊이 우선 탐색(DFS)**
현재 위치에서 가장 우선순위가 높은 노드 하나만 선택해 탐색해나가며, 더 이상 방문할 노드가 없을 경우 다시 이전의 노드로 되돌아와 다른 노드들을 탐색해가는 방법이다.

1. 처음 A 하나를 탐색한다.
2. A에 연결된 노드는 B와 C 두 개이며, 이 중 우선순위가 높은 B를 탐색한다.
3. B에 연결된 노드는 D와 E 두 개이며, 이 중 우선순위가 높은 D를 탐색한다.
4. D에 연결된 노드는 G 하나이므로, G를 탐색한다.
5. G에 연결된 노드는 E와 F이므로, 이 중 우선순위가 높은 E를 탐색한다.
6. E에 연결된 노드는 B와 C 두 개인데, B는 이미 탐색한 노드이므로 무시하고 C를 탐색한다.
7. C에서 더 이상 방문할 새 노드가 없기 때문에 이전인 E로 되돌아간다.
E에서도 더 이상 방문할 새 노드가 없기 때문에 G로 되돌아간다.
G에 연결된 노드 중 탐색하지 않았던 F를 탐색한다.

결과) **A-B-D-G-E-C-F**

◈ **너비 우선 탐색(BFS)** 방법 1)
현재 위치에서 인접한 이웃 노드를 모두 탐색한 뒤, 다음에 그 이웃 노드들의 이웃 노드들을 탐색해나가는 방법이다.

1. 처음 A를 탐색한다.
2. A의 이웃노드인 B와 C 중, 우선 순위가 높은 B를 먼저 탐색한다.
3. 그리고 C를 탐색한다. 이로써 A의 이웃노드는 모두 탐색이 끝났다.
4. 이번엔 B의 이웃 노드 D와 E 중 우선순위가 높은 D를 먼저 탐색한다.
5. 그리고 E를 탐색한다. 이로써 B의 이웃노드는 모두 탐색이 끝났다.
6. C의 이웃 노드는 더 이상 탐색할 것이 없으므로 다음으로 D의 이웃 노드인 G를 탐색한다.
7. E의 이웃 노드는 더 이상 탐색할 것이 없으므로 다음으로 G의 이웃 노드인 F를 탐색한다.

◈ **너비 우선 탐색(BFS)** 방법 2)
노드를 방문해나갈 때마다 그 이웃 노드들을 대기열 뒤에 등록하는 것으로 생각하면 된다.

1. 대기열을 생성해 시작 지점인 A를 등록한다.
2. A를 방문하고 이웃 노드인 B, C를 등록한다.
-> **A-B-C**
3. B를 방문하고 이웃 노드인 D, E를 등록한다. 이 때 B의 뒤가 아닌, 전체 대기열 뒤에 등록한다.
-> **A-B-C-D-E**
4. C를 방문하고 이웃 노드인 E는 이미 등록되어 있으므로 건너 뛴다.
5. D를 방문하고 이웃 노드인 G를 등록한다.
-> **A-B-C-D-E-G**
6. E를 방문하고 이웃 노드인 G는 이미 등록되어 있으므로 건너 뛴다.
7. G를 방문하고 이웃 노드인 F를 등록한다.
-> **A-B-C-D-E-G-F**
8. F를 방문한다.

문 7. 다음 후위(postfix) 표기식을 전위(prefix) 표기식으로 바꾼 것은? (단, 표기식에서 +, -, *, /는 연산자이고 A, B, C, D, E는 피연산자이다)

A B C * D / + E -

① - + A * / B C D E
② - / * + A B C D E
③ + / * - A B C D E
④ - + A / * B C D E

답 ④

1) A B C * D / + E - => A * B C D / + E -
2) A * B C D / + E - => A / * B C D + E -
3) A / * B C D + E - => + A / * B C D E -
4) + A / * B C D E - => - + A / * B C D E

문 8. 가상 기억장치 기술에 대한 설명으로 옳지 않은 것은?

① 가상 주소(virtual address)에서 물리 주소(physical address)로의 주소 변환(address translation)이 이루어진다.
② 가상 주소와 물리 주소의 비트 수가 서로 다를 수 있다.
③ 다중 프로그래밍 정도(degree of multiprogramming)가 높아짐에 따라 CPU 이용률(utilization)은 계속 높아진다.
④ 서로 다른 프로세스가 동일한 물리 기억장치 영역을 공유할 수 있다.

답 ③

③ 다중 프로그래밍 초반에는 CPU가 낭비되는 시간이 없이 계속 프로세스를 처리하기 때문에 CPU 이용률이 높아진다. 하지만 계속 다중 프로그래밍 정도를 높이게 되면 주기억장치에 없는 데이터를 요구하는 횟수가 계속 증가하게 되고, 나중엔 프로세스가 데이터를 처리하는 시간보다 페이지 교체에 더 많은 시간을 허비하게 되어 결국 CPU 이용률이 현저히 떨어지게 되는데 이러한 현상을 **스래싱**(thrashing)이라고 한다.
비슷한 말로 **멀티 스레딩**(multithreading)이 있는데, 멀티 스레딩이란 컴퓨터 제어 흐름의 최소 단위인 스레드를 병행 처리하는 것을 의미한다.

<오답 체크> ① CPU는 가상 저장장치에 있는 데이터를 직접 처리할 순 없다. 반드시 주기억장치로 불러온 뒤 실행해야 하는데, 이때 실행 중인 프로세스의 가장 주소(디스크에 생성되는 주소)를 실제 물리 주소로 변환하는 작업이 필요하다. 이렇게 프로세스 실행 중에 가상 주소를 실제 주소로 바꾸는 작업을 **동적 주소 변환(DAT** : Dynamic Address Translation)이라고 한다.
② 가상 기억장치와 주기억장치의 크기는 다를 수 있기 때문에 주소의 비트 수 또한 다를 수 있다.
④ 두 개 이상의 프로세스를 병행 처리하게 되면, 데이터인 물리 기억장치 영역도 공유하게 된다.

문 9. 다음 2 진 표현이 나타내는 IEEE 754 표준 단정도 (single precision) 부동소수점 수의 값은?

11000001110101010000000000000000

① $+5.3125_{(10)}$
② $-26.625_{(10)}$
③ $+21.25_{(10)}$
④ $-13.3125_{(10)}$

답 ②

IEEE 754 표준 단정도는
$(-1)^S \times 1.F \times 2^E$ 로 표현된다.
S는 부호 1비트, E는 지수부 8비트, F는 가수부 23비트
단 지수부는 127 바이어스(bias)법을 따른다.

1 / 10000011 / 10101010000000000000000
앞의 1비트는 부호 비트 1이기 때문에 음수(-)
중간 8비트는 지수 비트이며, 127 바이어스를 따르기 때문에 10000011에서 127을 빼면, 00000100(= 4)이 된다.
뒤에 23비트는 가수 비트이며, 1010101이다.
따라서 문제의 값은
$- 1.1010101 \times 2^4 = -11010.101_{(2)}$
10진수로 바꾸면 -26.625가 된다.

문 10. UNIX 에서의 프로세스 간 통신(interprocess communication)에 대한 설명으로 옳지 않은 것은?

① 세마포어(semaphore) 동작은 중단될 수 없는 원자성을 가진다.
② 시그널(signal)은 커널 혹은 프로세스가 다른 프로세스에게 비동기적으로 특정 사건을 통지하는 데 사용된다.
③ 지명 파이프(named pipe)를 통해 통신하는 프로세스 간에는 부모.자식 관계가 요구된다.
④ 공유메모리(shared memory)에 대한 상호 배제(mutual exclusion)는 운영체제가 보장하지 않는다.

답 ③

③ 유닉스에서 **파이프**(pipe)란 한 프로세스에서 다른 프로세스로 정보를 전달하는 통로 또는 기술로, 한 프로세스가 쓰고 다른 프로세스가 읽는 선입선출 형태의 큐(Queue)라 할 수 있다.
어떤 프로세스가 파이프에 데이터를 기록하려고 할 때 파이프 내 공간이 남아있다면 기록이 수행되지만, 공간이 부족하다면 그 프로세스는 차단된다. 즉 한 순간에 1개의 프로세스만이 파이프에 접근할 수 있고, 상호배제를 수행한 것이 된다.
파이프에는 익명 파이프(anonymous pipe)와 지명 파이프(named pipe)가 있는데, **익명 파이프는 서로 관련된 프로세스들만 공유할 수 있고, 지명 파이프는 관련이 없는 프로세스들도 공유할 수 있다.**
또한 일반 파이프는 사용하는 프로세스가 실행 중에만 존재하지만, 지명 파이프는 영구 프로세스가 소멸해도 계속 존재하기 때문에 사용하지 않으면 수동으로 제거할 필요가 있다.

<오답 체크> ① **세마포어**(semaphore)는 운영 체계 또는 프로그램 처리에서 공유 자원에 대한 접속을 제어하기 위해 변수를 사용하는 방법이다. 세마포어는 P와 S 두 개의 변수에 의해 작동하며, P는 임계 구역에 들어가기 전에 수행되고, V는 임계 구역에서 나올 때 수행된다. 이때 변수 값을 수정하는 연산은 모두 원자성을 만족해야 하며, 한 프로세스(또는 스레드)에서 세마포어 값을 변경하는 동안 다른 프로세스가 이 값을 변경해서는 안 된다.

② 유닉스에서 **Signal**(시그널)은 프로세스에게 어떤 상황이 발생했음을 알리기 위한 인터럽트 신호이다. 특정 상황에서만 필요에 의해 발생하기 때문에 비동기식 틍보 방식이다.

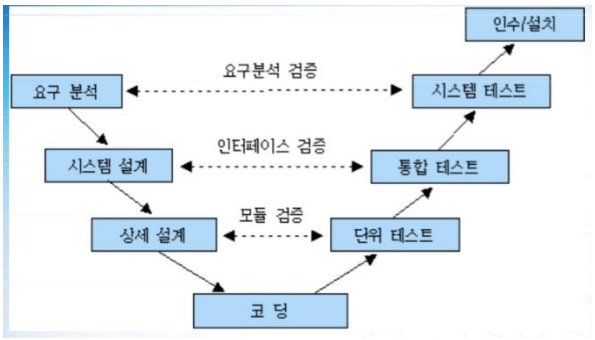
④ 유닉스에서 상호배제를 보장하지 않기 때문에 프로그래밍을 할 때 세마포어 등의 방식을 이용해 교착상태에 대한 대비를 해야 한다.

문 11. 폭포수(waterfall) 모델의 변형으로 산출물보다는 각 개발 단계의 테스트에 중점을 두며, 테스트 활동이 분석 및 설계와 어떻게 관련되어 있는지 보여 주는 소프트웨어 개발 모델은?

① 나선형(spiral) 모델
② 단계적 개발(phased development) 모델
③ 원형(prototyping) 모델
④ V 모델

답 ④

④ V모델은 폭포수 모델의 확장된 형태로, 각 단계별 시스템 검증, 테스트 작업을 강화한 모델이다.



<오답 체크> ① 나선형(spiral) 모델
각 단계별 나선 모양을 그려며 반복 수행하는 모델이다.
1. 요구 사항 및 위험 분석
2. 계획 및 건축가 반복
3. 구현
4. 테스트 및 확인

② 단계적 개발(phased development) 모델
전체 시스템을 여러 개의 버전으로 나누고 각 버전을 순차적으로 개발하는 모델이다. 개발자가 먼저 릴리스 1을 개발하여 사용자에게 제공하면 사용자가 개발된 릴리스 1을 사용하고, 그 동안 개발자가 다음 버전인 릴리스 2를 개발한다. 이처럼 개발과 사용을 병행하는 과정을 반복하여 진행하면서 완료한다.(게임에서 패치를 내놓는 것과 비슷한 개념으로 볼 수 있다.) 따라서 중요한 기능은 첫 번째 버전에 포함하는 것이 좋다.

③ 프로토타이핑(prototyping) 모델(= 원형 모델)
요구 분석이 정확하지 않을 때, 개발의 일부분만을 원형(프로토타입, prototype)으로 개발하여 사용자에게 제공하여 시험 사용하게 한 뒤, 이를 통해 요구를 분석하고 점검·평가·개선 작업을 통해 개발하는 모델이다.
프로토타입은 실험적(experimental) 프로토타입과 진화적(evolutionary) 프로토타입이 있다.
실험적(experimental) 프로토타입은 사용자의 요구를 알아내기 위해서만 사용할 뿐, 최종 프로토타입을 바탕으로 실제 제품을 만들고, 프로토타입은 버린다.
진화적(evolutionary) 프로토타입은 버리지 않고, 지속적으로 개선·보완하여 최종 시스템으로 완성시킨다.

문 12. 중앙처리장치와 주기억장치 사이에 있는 기억장치로서, 둘 사이의 속도 차이로 인한 컴퓨터 시스템 성능 저하를 경감하기 위한 것은?

① 캐시 기억장치
② 보조 기억장치
③ ROM
④ 레지스터

답 ①

◈ 기억장치 접근 속도
CPU(레지스터) > 연관 메모리 > 캐시 메모리 > 주기억장치(RAM) > 보조기억장치

<오답 체크> ③ **ROM**(롬, Read Only Memory)
읽기 전용 메모리. 컴퓨터에 설치되기 전에 데이터가 특수한 방법으로 기록되어 있는데, 이것은 전원을 넣었을 때의 동작 개시 프로그램이나 기본 동작 소프트웨어(BIOS) 등이 들어 있다.

문 13. 관계형 데이터베이스 언어인 SQL에 대한 설명으로 옳은 것은?

① 데이터 정의어(DDL)를 이용하여 데이터를 검색한다.
② 데이터 조작어(DML)를 이용하여 권한을 부여하거나 취소한다.
③ DELETE 문은 테이블을 삭제하는 데 사용한다.
④ SELECT 문에서 FROM 절은 필수 항목이고, WHERE 절은 선택 항목이다.

답 ④

④ SELECT 문의 기본적인 형태는 다음과 같은데, 이 때 WHERE 조건절은 생략이 가능하다.

SELECT [열] FROM [테이블] WHERE [조건]

<오답 체크> ①② **DDL**(Data Definition Language, 데이터 정의어)은 스키마나 데이터 구조를 정의, 변경, 삭제하는데 사용되는 명령어들이다.
생성(CREATE), 변경(ALTER), 제거(DROP), 이름 정정(RENAME), 완전 제거(TRUNCATE) 등
DML(Data Manipulation Language, 데이터 조작어)은 데이터를 입력, 검색, 수정하는 데 사용되는 명령어들이다.
추가(INSERT), 수정(UPDATE), 검색(SELECT), 삭제(DELETE) 등
DCL(Data Control Language, 데이터 제어어)은 데이터베이스에 접근하거나 객체에 권한을 주는 데 사용하는 명령어들이다.
권한의 부여(GRANT) 및 박탈(REVOKE), 트랜잭션 저장(COMMIT) 및 취소(ROLLBACK) 등
이 중 트랜잭션을 제어하는 COMMIT와 ROLLBACK만을 분리해 **TCL**(Transaction Control Language)라고 부르기도 한다.
③ 데이터를 삭제하는 데는 DELETE 문을 쓰지만, 스키마, 도메인, 뷰, 인덱스를 삭제하는 데는 DROP 문을 쓴다.

문 14. 삽입 정렬을 사용하여 자료를 오름차순으로 정렬한다. 초기 및 2회전 후의 자료가 다음과 같다면 4회전 후의 결과는?

초기 자료 : 69, 30, 10, 2, 16, 8, 31, 22
2회전 후의 자료 : 10, 30, 69, 2, 16, 8, 31, 22

① 2, 10, 16, 30, 69, 8, 31, 22
② 8, 2, 10, 30, 16, 69, 22, 31
③ 16, 2, 10, 30, 69, 8, 22, 31
④ 2, 10, 30, 69, 16, 8, 31, 22

답 ①

삽입 정렬(Insertion Sort)은 배열을 정렬된 앞 부분과 정렬되지 않은 뒷 부분으로 나눠, 이미 정렬된 앞 부분 배열과 비교하여 **뒷 부분의 요소를 하나씩 적당한 위치를 찾아 삽입**하는 방법이다. 시간복잡도는 $O(n^2)$이다.

초기 [69 | 30 | 10 | 2 | 16 | 8 | 31 | 22]
1회전 : 먼저 앞의 1, 2번 요소를 정렬한다.
[30 | 69 | 10 | 2 | 16 | 8 | 31 | 22]
2회전 : 3번 요소(10)를 앞의 정렬된 배열의 적당한 위치를 골라 삽입한다.
[10 | 30 | 69 | 2 | 16 | 8 | 31 | 22]
3회전 : 4번 요소(2)를 앞의 정렬된 배열의 적당한 위치를 골라 삽입한다.
[2 | 10 | 30 | 69 | 16 | 8 | 31 | 22]
4회전 : 5번 요소(16)를 앞의 정렬된 배열의 적당한 위치를 골라 삽입한다.
[2 | 10 | 16 | 30 | 69 | 8 | 31 | 22] <-- ①
5회전 : 6번 요소(8)를 앞의 정렬된 배열의 적당한 위치를 골라 삽입한다.
[2 | 8 | 10 | 16 | 30 | 69 | 31 | 22]
6회전 : 7번 요소(31)를 앞의 정렬된 배열의 적당한 위치를 골라 삽입한다.
[2 | 8 | 10 | 16 | 30 | 31 | 69 | 22]
7회전 : 8번 요소(22)를 앞의 정렬된 배열의 적당한 위치를 골라 삽입한다.
[2 | 8 | 10 | 16 | 22 | 30 | 31 | 69]

삽입정렬에서 n회전 후의 결과는 1번부터 n+1번까지 요소를 정렬한 것이 된다. 문제에서 4회전 결과를 물어봤기 때문에, 앞에서부터 5번째 요소인 16까지는 정렬하고, 8 이후는 그대로 둔 것을 고르면 된다.

문 15. 다음 Java 언어로 작성한 프로그램의 실행 결과는?

```
public class Test {
    public static void main(String[] args) {
        int ar[] = {10, 20, 30, 40, 50};
        int sum = 0, a = 100, b = 0;
        try {
            for(int i = 0; i < ar.length; i++) {
                sum += ar[i];
            }
            System.out.println(sum);
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Array Index Out Of Bounds Exception");
        }
        try {
            float z = a / b;
            System.out.println(z);
        } catch (ArithmeticException e) {
            System.out.println("Arithmetic Exception");
        }
    }
}
```

① 100
0.0
② 100
Array Index Out Of Bounds Exception
③ 150
Arithmetic Exception
④ 150
/ by zero at Test.main(Test.java:14)

답 ③

◈ try~catch~finally 문은 먼저 try 블록에서 프로그래밍을 수행시킨다. 이 때 try 블록에서 오류가 발생하게 되면 try 블록은 수행을 멈추고, catch 블록으로 이동한다.
catch 블록에서는 try 블록에서 발생한 오류에 따라 catch 블록이 실행된다.
(아직까지는 catch 블록에서 하나의 오류만 처리하도록 설정되어 출제되었는데, 앞으로는 오류 종류를 구별하는 방식으로 출제될지도 모른다.)
finally 블록은 앞에서 오류 발생 및 실행 여부와 관계없이 최종적으로 실행되는 블록이다.
catch 블록과 finally 블록은 생략이 가능하다.

▷ 첫 번째 try문은 배열 ar[]의 요소값을 모두 더하는 연산을 수행한 뒤 sum 값을 출력한다.
따라서 10 + 20 + 30 + 40 + 50 = 150 이 출력된다.
catch문의 ArrayIndexOutOfBoundsException 오류는 읽을 수 없는 배열 값을 불러오려 할 때 발생하는 오류인데, try문에서 오류가 발생하지 않았기 때문에 건너뛴다.

▷ 두 번째 try문에서 float z = a / b; 에서 b값이 0이기 때문에 나눗셈 연산 오류 ArithmeticException 오류가 발생한다.
따라서 try문은 중단하고 catch문을 수행하여 'Arithmetic Exception'이라는 문구가 출력된다.

문 16. TCP 헤더에 포함된 필드에 대한 설명으로 옳은 것만을 모두 고른 것은?

> ㄱ. 송신지(source) 포트 번호는 송신지 응용 프로그램에 할당된 포트 번호이다.
> ㄴ. 확인 응답 번호(acknowledgment number)는 성공적으로 수신한 데이터의 첫 바이트에 부여된 순서 번호(sequence number)이다.
> ㄷ. 플래그(flags)는 TCP 동작 제어를 위해 사용되는 1비트 크기의 SYN, ACK 등으로 이루어진다.
> ㄹ. 윈도우 크기(window size)는 송신 측에서 송신할 수 있는 비트 단위의 최대 데이터 크기를 나타낸다.

① ㄱ, ㄴ
② ㄱ, ㄷ
③ ㄷ, ㄹ
④ ㄴ, ㄹ

답 ②

ㄱ. 서버가 제공하는 여러 프로그램 중 어느 프로그램이 요청한 서비스인지를 구분하기 위하여 **송신지 포트**(Source Port) 번호가 존재한다. 어떤 응용 프로그램의 요청인지 확인하여, 데이터를 응용 프로그램에 맞게 처리할 수 있다.

ㄷ. TCP 플래그 비트는 각각 1비트 크기를 가지는 6개의 제어 플러그 필드가 있으며,(도합 6비트) 이들은 논리적인 TCP 연결화선 제어 및 데이터 관리를 위해 사용한다.
(이후 버전업을 통해 3개가 추가되어 현재는 9비트로 되어 있다.)
6개의 플래그는 **URG**(Urgent), **ACK**(Acknowledgement), **PSH**(Push), **RST**(Reset), **SYN**(Synchronize), **FIN**(Finish) 패킷이다.

<오답 체크> ㄴ. 확인 응답 번호(acknowledgment number)는 상대방으로부터 수신한 바이트의 '마지막 순서 번호+1'이다. 송신 측에 데이터를 잘 받았다는 것을 알려주고, 다음으로 전송해주길 기다리고 있는 바이트의 순서번호를 의미한다.
6번 바이트까지 정상적으로 수신하였다면, 다음 7번을 보내달라는 의미로 확인응답번호 7을 전송한다.

ㄹ. 윈도우의 크기는 수신 측에서 수신할 수 있는 크기를 나타낸다.

문 17. 함수 수행을 위한 정보가 저장되는 프로세스 메모리 영역은?

① 데이터(data) 영역
② 힙(heap) 영역
③ 스택(stack) 영역
④ 텍스트(text) 영역

답 ③

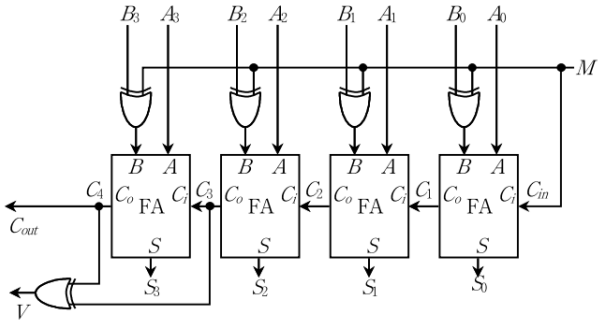
③ **Stack**(스택) 영역
함수 호출 시 생성되는 지역 변수와 매개 변수가 저장되는 영역
함수 호출이 완료되면 사라짐
쓰기 가능, 크기 가변

<오답 체크> ① **Data**(데이터) 영역
전역 변수와 정적(static) 변수가 할당되는 영역
프로그램의 시작과 동시에 할당되고, 프로그램이 종료되어야 메모리에서 소멸됨
쓰기 가능, 크기 고정
(좀 더 세분화하면 데이터 영역에는 초기화된 전역 변수와 정적 변수가 저장되고, 초기화되지 않은 전역 변수와 정적 변수는 BSS (Block Stated Symbol) 영역에 저장된다.)

② **Heap**(힙) 영역
필요에 의해 동적으로 메모리를 할당 할 때 사용
쓰기 가능, 크기 가변

④ **Text**(텍스트) 영역 = Code(코드) 영역
소스 코드 자체를 기계어로 변환하여 저장되는 영역으로, 변수가 아닌 순수 코드만 있는 공간이다.
쓰기 금지, 크기 고정

문 18. 전가산기(FA: Full Adder)는 두 입력 A, B 및 입력캐리 C_i를 더해서 합 S와 출력캐리 C_o를 만들어 내는 회로이다. 4 개의 전가산기를 사용한 다음 연산기에서 오버플로우(overflow)가 발생한 경우가 아닌 것은?



① M = 0, C_{out} = 1, 부호 없는(unsigned) 연산으로 해석
② M = 1, C_{out} = 1, 부호 없는 연산으로 해석
③ M = 0, V = 1, 2 의 보수를 사용하는 부호 있는(signed) 연산으로 해석
④ M = 1, V = 1, 2 의 보수를 사용하는 부호 있는 연산으로 해석

답 ②

이해하기 편하도록 정답 오답 순서가 아닌,
부호 없는 연산, 부호 있는 연산 순서로 설명을 하겠다.

★ 부호 없는 연산의 경우에 대한 설명 ★

① 부호 없는 연산이기 때문에, 입력 A와 B는 모두 양수로 취급한다. 그런데 마지막 C_{out}(캐리)값이 1이 나왔다는 건 현재 비트값을 초과하는 **자리 올림수가 발생했다는 의미이며, 따라서 오버플로우에 해당**한다.

▶ ex) 1100 (12) + 0111 (7) = 10011 (19) 오버플로우 발생

4자리 2진수는 최대 15까지만 표현이 가능한데, 덧셈 결과 15가 넘어 자리올림수가 발생하였다. 오버플로우

② M이 1일 경우에는 B의 각 비트값들은 반전되어서 입력되고 초기 C_{in}값 1이 추가로 입력된다. 따라서 **2의 보수에 의한 뺄셈 계산**이 된다.(A-B)
부호 없는 연산이기 때문에 음수값은 불가능하기 때문에, A가 B보다 커야 오버플로우가 발생하지 않는다.

▶ ex 1) 1100 (12) - 0101 (5) 이것을 2의 보수의 방법으로 계산하면, 1100 + 1011 = 10111 이 된다. (C_{out} 출력캐리 발생)

앞의 출력캐리 C_{out} 값을 버리면 0111 (7)로 정상적인 값이 나온다.
이처럼 **2의 보수에 의한 부호 없는 뺄셈 연산에서는 캐리가 발생해야 정상**인 것이다.

▶ ex 2) 만일 B가 A보다 크다면, 결과값이 음수로 나오므로 부호 없는 연산으로 불가능한 계산이 된다.

0101 (5) - 1100 (12) = 0101 + 0100 = 1001

불가능한 계산을 하였더니 앞에 캐리값이 없다.
따라서 **2의 보수에 의한 부호 없는 뺄셈 연산의 경우 오히려 캐리가 발생하지 않으면 오버플로우가 발생**하는 것이다.

★ **2의 보수 부호 있는 연산의 경우에 대한 설명** ★

V=1이라는 것은 마지막 네 번째 자리올림수(C_4)와 그 앞 세 번째 자리올림수(C_3)의 값이 다른 경우를 의미한다.(XOR 회로를 이용한다.)

오버플로우는 표현 가능한 수의 범위를 넘는 것이기 때문에, 다음의 두 경우에 오버플로우가 발생한다.

(가) 양수와 양수를 더했는데(또는 양수에서 음수를 뺐는데) 음수가 나오는 경우

(나) 음수와 음수를 더했는데(또는 음수에서 양수를 뺐는데) 양수가 나오는 경우

부호가 다른 값을 더하거나, 부호가 같은 값을 뺄 경우에는 오버플로우가 발생하지 않는다.

(또는 연산기 상에서 오버플로우는 부호비트가 같은 것을 더했는데, 부호비트가 바뀌는 것을 보고 오버플로우로 판정을 내리게 된다. 이 부분은 설명이 길므로 각자 생각해보자.)

③ M이 0이기 때문에 덧셈을 하는 경우이다.

▶ ex1) 우선 양수와 음수를 더하는 경우

A = 0011 (+3), B = 1010 (-6)

부호가 다른 값을 더할 경우는 오버플로우가 발생하지 않을 것이다.

0011 + 1010 = 1101 (-3)

옳은 계산이 나왔다.(**오버플로우 X**)

C_4 = 0, C_3 = 0, **V = 0**

▶ ex2) 양수와 양수를 더하는 경우 = **(가)**의 경우

A = 0110 (+6), B = 0101 (+5)

A와 B를 더하면 +11이 나오는데, 4비트의 부호 있는 연산이기 때문에 표현 가능한 최댓값은 +7이므로 계산이 불가능하다. 오버플로우가 발생할 것이다.

0110 + 0101 = 1011 (-5)

부호비트가 1이므로 음수인데, 양수끼리 더했는데 음수가 나왔기 때문에 잘못된 계산이며, **오버플로우가 맞다.**

C_4 = 0, C_3 = 1, **V = 1**

▶ ex3) 음수와 음수를 더하는 경우 = **(나)**의 경우

A = 1101 (-3), B = 1010 (-6)

A와 B를 더하면 -9가 나오는데, 4비트의 부호 있는 연산이기 때문에 표현 가능한 최솟값은 -8이므로 계산이 불가능하다. 오버플로우가 발생할 것이다.

1101 + 1010 = 10111 (-5)

부호비트가 0이므로 양수인데, 음수끼리 더했는데 양수가 나왔기 때문에 잘못된 계산이며, **오버플로우가 맞다.**

C_4 = 1, C_3 = 0, **V = 1**

따라서 **2의 보수를 사용하는 부호있는 연산에서는 M = 0일 때, V = 1이면 오버플로우**인 것을 확인할 수 있다.

④ M = 1이기 때문에 2의 보수를 이용한 뺄셈을 하는 경우이다.

▶ ex1) 음수에서 음수를 빼는 경우

A = 1101 (-3), B = 1010 (-6)

부호가 같은 값을 뺄 경우는 오버플로우가 발생하지 않을 것이다.

1101 - 1010 = 1101 + 0110 = 10011 (+3)

옳은 계산이 나왔다.(**오버플로우 X**)

C_4 = 1, C_3 = 1, **V = 0**

▶ ex2) 양수에서 음수를 빼는 경우 = **(가)**의 경우

A = 0110 (+6), B = 1100 (-4)

A에서 B를 빼면 +10이 나오는데, 표현 가능한 최댓값은 +7이므로 계산이 불가능하다. 오버플로우가 발생할 것이다.

0110 - 1100 = 0110 + 0100 = 1010 (-6)

부호비트가 1이므로 음수인데, 양수에서 음수를 뺐는데 음수가 나왔기 때문에 잘못된 계산이며, **오버플로우가 맞다.**

C_4 = 0, C_3 = 1, **V = 1**

▶ ex3) 음수에서 양수를 빼는 경우 = **(나)**의 경우

A = 1001 (-7), B = 0010 (+2)

A에서 B를 빼면 -9가 나오는데, 최솟값은 -8이므로 계산이 불가능하다. 오버플로우가 발생할 것이다.

1001 - 0010 = 1001 + 1110 = 10111 (-5)

부호비트가 0이므로 양수인데, 음수에서 양수를 뺐는데 양수가 나왔기 때문에 잘못된 계산이며, **오버플로우가 맞다.**

C_4 = 1, C_3 = 0, **V = 1**

따라서 **2의 보수를 사용하는 부호있는 연산에서는 M = 1일 때, V = 1이면 오버플로우**인 것을 확인할 수 있다.

★★ 이를 정리하면

◈ 부호 없는 연산의 경우

M = 0 (덧셈), 출력캐리 발생 시(**C_{out} = 1**) 오버플로우

M = 1 (뺄셈) 출력캐리 미발생 시(**C_{out} = 0**)시 오버플로우

◈ 부호 있는 연산의 경우

덧셈 뺄셈 상관없이, 마지막 자리올림수(C_4)와 그 앞 자리올림수(C_3)가 다를 경우(**V = 1**), 오버플로우

문 19. 다음 C 프로그램의 실행 결과는?

```
#include 〈stdio.h〉

int func(int n) {
    if(n 〈= 1)
        return(n);
    else
        return(func(n - 1) + func(n - 2));

}
int main(void) {
    int n = 7;
    int i;
    int result = 0;

    for(i = 0; i 〈 n; i++)
        result += func(i);
    printf("%d", result);

    return(0);
}
```

① 0
② 12
③ 19
④ 20

답 ④

main 암수의 for문은 result에 func(0)부터 func(6)까지 더하는 연산을 수행한다.
(n=7일 때 0부터 6까지라는 것에 주의)

func(0) = 0
func(1) = 1
func(2) = func(1) + func(0) = 1 + 0 = 1
func(3) = func(2) + func(1) = 1 + 1 = 2
func(4) = func(3) + func(2) = 2 + 1 = 3
func(5) = func(4) + func(3) = 3 + 2 = 5
func(6) = func(5) + func(4) = 5 + 3 = 8

그리고 마지막 printf 문의 수행 결과 20이 출력된다.

◎ 참고로 위의 func 함수 알고리즘은 피보나치 수열 알고리즘이며, 프로그래밍 문제에서 자주 등장한다.

문 20. 운영체제가 프로세스(process)를 생성하는 과정을 순서대로 바르게 나열한 것은?

ㄱ. 새로운 프로세스를 위한 프로세스 식별자를 할당한다.
ㄴ. 새로운 프로세스를 스케줄링 큐의 준비 또는 준비/보류 리스트에 연결한다.
ㄷ. 새로운 프로세스를 위한 주소 공간과 프로세스 제어블록(process control block)을 할당한다.
ㄹ. 새로운 프로세스의 프로세스 제어블록을 초기화한다.

① ㄱ → ㄴ → ㄷ → ㄹ
② ㄱ → ㄷ → ㄹ → ㄴ
③ ㄷ → ㄹ → ㄱ → ㄴ
④ ㄷ → ㄹ → ㄴ → ㄱ

답 ②

ㄱ. 새로운 프로세스에 프로세스 식별자를 할당한다.
ㄷ. 새로운 프로세스의 구성 요소를 포함할 수 있는 주소 공간과 프로세스 제어블록(process control block)을 할당한다.
ㄹ. 프로세스 제어블록을 초기화한다.
프로세스 상태 정보, 프로그램 카운터, 스택 포인터 등의 초기화, 자원 요청, 프로세스 제어 정보(우선순위) 등이 포함된다.
ㄴ. 새로운 프로세스를 스케줄링 큐의 준비 또는 준비/보류 리스트에 연결한다.