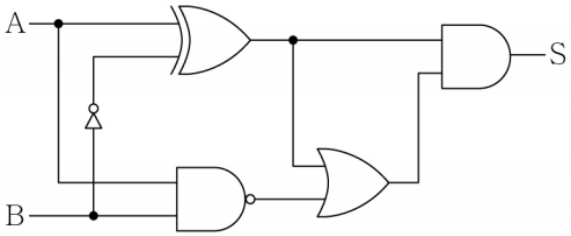
2016년 서울시 9급 컴퓨터일반 풀이

by 호이호이꿀떡

정답 체크

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

1. 다음 논리회로에서 A=1010, B=0010 일 때, S에 출력되는 값은?



① 1011 ② 1101
③ 0111 ④ 1110

답 ③

회로가 4개이므로, A와 B값에 대입하여 직접 실수 없이 빠르게 계산하는 것도 나쁘지 않은 방법이다.

굳이 위의 논리회로를 불 수식으로 표현하자면, 불 대수 흡수법칙을 알고 있으면 바로 구할 수 있다.
흡수법칙이란 A • (A+B) = A가 되는 법칙인데, 회로도의 오른쪽 부분만 잘라서 보면 흡수법칙과 동일한 형태인 것을 알 수 있다.
이를 이용해 위의 논리회로를 수식으로 표현하면,

(A⊕B` ) • (( A⊕B`)+AB) = A⊕B` 이 된다.

A = 1010, B` = 1101
따라서 S = A⊕B` = 0111

2. 현재 실행 중인 프로세스에 할당된 CPU사용권을 다른 프로세스에게 할당하려면, 현재 실행 중인 프로세스의 실행 정보를 저장하고 다음으로 실행할 프로세스의 실행정보를 가져오는 과정이 필요하다. 이 과정을 무엇이라고 하는가?

① 컨텍스트 스위칭(Context Switching)
② 가상메모리(Virtual Memory)
③ 교체정책(Replacement Strategy)
④ 디스패치(Dispatch)

답 ①

① 컨텍스트 스위칭(Context Switch, **문맥교환**)
하나의 프로세스가 CPU를 사용 중인 상태에서 다른 프로세스가 CPU를 사용하도록 하기 위해, 이전의 프로세스의 상태(문맥)를 보관하고 새로운 프로세스의 상태를 적재하는 작업을 말한다. 프로세스의 문맥은 프로세스 제어 블록(PCB)에 기록되어 있다.

<오답 체크> ② **가상메모리**(Virtual Memory)
실제 물리적인 메모리 공간 주기억장치(RAM) 외에 하드 디스크에 파일 형태로 따로 준비하는 가상의 메모리 공간으로, 부족한 시스템 메모리를 보조해주는 역할을 한다.

③ 페이지 교체 정책(Page Replacement Strategy)
CPU가 사용할 데이터가 주기억장치에 없는 상황을 **페이지 부재**(page fault)라고 한다. 페이지 부재 발생시 새로운 페이지를 할당하기 위해 현재 할당된 페이지 중 어느 것과 교체할지를 결정하는 정책을 의미한다.

④ 디스패치(dispatch)
프로세스의 상태 전이 중 하나로, 프로세스 실행 준비 리스트의 맨 앞에 있던 프로세스가 CPU를 점유하게 되는 것, 즉 준비(ready) 상태에서 실행(running) 상태로 바뀌는 것을 말한다.

▷ 블록(block)
실행(running) 상태에서 대기(waiting) 상태로 바뀌는 것

▷ 깨움(wakeup)
대기(waiting) 상태에서 준비(ready) 상태로 바뀌는 것

▷ 시간제한(timeout)
실행(running) 상태에서 준비(ready) 상태로 바뀌는 것

3. 다음 중 유효한 SQL 문장이 아닌 것은?

① SELECT * FROM Lawyers WHERE firmName LIKE '% and %';
② SELECT firmLoc, COUNT(*) FROM Firms WHERE employees < 100;
③ SELECT COUNT(*) FROM Firms WHERE employees < 100;
④ SELECT firmLoc, SUM(employees) FROM Firms GROUP BY firmLoc WHERE SUM(employees) < 100;

답 ④

④ 'group by'로 그룹화한 것에 조건을 추가하기 위해서는 'having' 문을 사용해야 한다.
SELECT firmLoc, SUM(employees) FROM Firms GROUP BY firmLoc HAVING SUM(employees) < 100;
Firms 테이블에서 레코드들을 FirmLoc 값이 같은 것들끼리 묶은 뒤, 각 묶음의 employees 값들을 합해서 100 미만일 경우에만 출력한다.(출력은 firmLoc 값과 해당 employees 값의 합 두 가지 속성을 출력한다.)

<오답 체크> ① SQL에서 where와 and를 이용해 조건을 두 개 적용하기 위해서는 각각 조건식을 써줘야 한다고 알고 있을 것이다.
예를 들어,
WHERE employees < 100 AND > 50 이라 쓰면 틀리고,
WHERE employees < 100 AND employees > 50 이라 써야 한다.(직원이 100 미만, 50초과인 레코드를 검색)

하지만 위 보기는 함정용으로 출제된 것으로 문자열을 검색할 때는 상황이 다르다.
WHERE문에서 LIKE 구문은 특정 단어가 포함된 것을 검색할 때 사용하는데, WHERE Student LIKE '박%' 이라면, Student 값이 '박'으로 시작하는 튜플을 검색한다.
문제에서처럼 WHERE FirmName LIKE '% and %'
으로 쓸 경우는 FirmName 값 중간에 'and' 단어가 들어간 레코드를 검색하겠다는 의미로, 여기서 and는 조건이 아니라, 문자열 'and'를 의미한다. 따라서 오류가 발생하지 않는다.

②③ Firms 테이블로부터 employees 값이 100 미만인 레코드(튜플)의 개수를 반환해준다.
②번의 경우 출력 속성값이 firmLoc과 Count(*) 두 가지인데, 그룹화해서 묶지 않았기 때문에 그냥 employees 값이 100 미만인 레코드 수를 반환한다.

4. 다음 중 나머지 셋과 역할 기능이 다른 하나는?

① Array processor
② DMA
③ GPU
④ SIMD

답 ②

② **DMA**(Direct Memory Access)는 주변장치들(하드디스크, 그래픽 카드, 네트워크 카드, 사운드 카드 등)이 중앙처리장치(CPU)의 도움 없이 메모리에 직접 접근하여 데이터를 읽거나 쓸 수 있도록 하는 기능이다. DMA가 지원되면 CPU가 데이터 전송에 관여하지 않아도 되므로 컴퓨터 성능이 좋아진다.

▶ DMA를 제외한 나머지 셋은 컴퓨터의 병렬 처리 기법이나 프로세서들이다.
DMA가 CPU 대신 데이터를 주고 받는 역할을 수행하는 것을 병렬 처리라고 생각할 수도 있지만, DMA는 데이터를 전송하는 역할만 할 뿐 전송된 데이터를 처리하는 건 CPU가 담당하기 때문에 데이터 병렬 처리에는 해당하지 않는다.

<오답 체크> ④ **SIMD**(Single instruction stream, multiple data streams)
한 번에 데이터 여러 개를 명령어 하나로 처리하는 기법이며, SIMD 프로세서는 병렬적으로 동작하는 다중 처리 장치를 가진 프로세서를 의미한다. SIMD 기법은 벡터 프로세서(배열 프로세서)에서 많이 사용되며 비디오 게임 콘솔이나 그래픽 카드와 같은 멀티미디어 분야에 자주 사용된다.

① **벡터 프로세서**(Vector processor) 또는 **배열 프로세서**(Array processor)란, 컴퓨터에서 벡터는 1차원 배열의 데이터를 의미하는데, 이 벡터라고 불리는 다수의 데이터를 병렬적으로 처리하는 명령어를 가진 CPU를 말한다.

③ **GPU**(Graphic Processing Unit)는 컴퓨터의 영상정보를 처리하거나 화면 출력을 담당하는 그래픽카드를 말한다. 중앙처리장치(CPU)의 그래픽 처리 작업을 돕기 위해 만들어졌으며, CPU의 그래픽 작업으로 인해 생기는 병목 현상을 해결하기 위해 개발되었다. GPU를 그래픽 가속기(Graphics Accelerator)라고도 하며, CPU보다 병렬 처리 성능이 더욱 뛰어나다.
고품질의 이미지·영상 정보를 처리하는 데 최적화되어 있기 때문에 특히 연산 능력이 매우 뛰어나며, 최근에는 빅데이터와 3D 그래픽 데이터 등이 늘고, 특히 최근 유행하고 있는 가상화폐(비트코인 등) 채굴에도 CPU보다 GPU가 더욱 뛰어난 성능을 보이면서 CPU보다 GPU가 점점 더 주목받고 있다.

5. 다음은 IPv6에 대한 설명이다. 옳지 않은 것은?

① 기존의 IP 주소 공간이 빠른 속도로 고갈되어 왔기 때문에 고안되었다.
② IPv6는 IP 주소 크기를 기존의 4바이트에서 6바이트로 확장했다.
③ IPv6는 유니캐스트, 멀티캐스트 주소뿐만 아니라 새로운 주소 형태인 애니캐스트 주소가 도입되었다.
④ 네트워크 프로토콜을 바꾼다는 것은 매우 어렵기 때문에 IPv6로의 전환을 위해 여러 방법들이 고안되었다.

답 ②

② IPv4는 8비트씩 4자리, 총 32비트(4바이트)
IPv6는 16비트씩 8자리, 총 128비트(16바이트)

<오답 체크> ③ IPv6에는 기존 IPv4에 있던 브로드캐스트 주소 방식 대신 애니캐스트 주소 방식이 도입되었다.(유니캐스트, 멀티캐스트 주소 방식은 동일)

▷ 유니캐스트 주소(Unicast Address) 1 대 1로 데이터 전송
▷ 멀티캐스트 주소(Multicast Address) 1 대 다수로, 특정한 여러 노드로 데이터 전송
▷ 브로드캐스트 주소(Broadcast Address) 1 대 다수로, 불특정한 여러 노드로 또는 해당 네트워크의 모든 노드로 데이터를 전송 (IPv4에서만 사용)
▷ 애니캐스트 주소(Anycast Address) 같은 서비스를 하는 여러 개의 서버가 같은 애니캐스트 주소를 가질 수 있으며, 가장 효율적으로 서비스할 수 있는 또는 가장 근접한 서버가 데이터를 전송 (IPv6에서만 사용. 일부 IPv4에서 가능하기는 하나, 시험 이론에서는 IPv6에서 도입된 것으로 알고 있자.)

◈ IP 주소 고갈 문제를 해결하기 위한 방법
▷ IPv6
▷ NAT(Network Address Translation): 공인 IP 주소를 다수가 함께 공유하여 사용
▷ DHCP(Dynamic Host Configuration Protocol, 동적 호스트 구성 프로토콜)

◈ IPv4와 IPv6 전환 기술
▷ 터널링(Tunneling): IPv4 망에 터널을 만들어 IPv6가 지나갈 수 있게 하는 기술
▷ 듀얼 스택(Dual Stack): 하나의 시스템에서 IPv4와 IPv6프로토콜을 함께 처리
▷ 헤더 변환(Header Translation) 패킷의 헤더 변환

6. 다음 정렬 알고리즘 중 최악의 경우에 시간복잡도가 가장 낮은 것은?

① 버블 정렬(Bubble sort)
② 삽입 정렬(Insertion sort)
③ 퀵 정렬(Quick sort)
④ 힙 정렬(Heap sort)

답 ④

④ 힙 정렬만 최악의 경우 시간 복잡도가 $O(n\log_2 n)$이고, 나머지는 $O(n^2)$이다.

<오답 체크> ③ 퀵 정렬의 경우 평균 시간 복잡도는 $O(n\log_2 n)$이지만, 최악의 경우 $O(n^2)$이다.

◈ 정렬 알고리즘 시간복잡도(Time complexity)

알고리즘	최선	평균	최악
삽입 정렬	$O(n)$	$O(n^2)$	$O(n^2)$
선택 정렬	$O(n^2)$	$O(n^2)$	$O(n^2)$
버블 정렬	$O(n^2)$	$O(n^2)$	$O(n^2)$
셸 정렬	$O(n)$	$O(n^{1.5})$	$O(n^2)$
퀵 정렬	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(n^2)$
힙 정렬	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(n\log_2 n)$
합병 정렬	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(n\log_2 n)$
기수 정렬	$O(dn)$	$O(dn)$	$O(dn)$

7. 다음 C 프로그램의 실행 결과는?

```
#include <stdio.h>
int main()
{
    char* array1[2]={"Good morning", "C language"};
    printf("%s ₩n", array1[0]+5);
    printf("%c ₩n", *(array1[1]+6));
    return 0;
}
```

① Good morning
C-language
② morning
a
③ morning
g
④ morning
u

답 ④

char* array1[2]={"Good morning", "C language"};
이렇게 배열을 선언하였는데, 문자열을 배열로 선언하였을 경우, 해당 값이 들어간다고 생각하지 말고, 해당 값을 가리키는 것이라고 생각해야 한다.
array1[0]는 "Good morning"의 시작 지점인 'G'를 가리키고, array1[1]는 "C language"의 시작 지점인 'C'를 가리킨다.

첫 번째 printf문에서 '%s'는 문자열 스트림을 출력한다는 의미이다. 문자열 스트림을 출력할 때는 해당 위치부터 끝까지 전부 출력한다.
첫 번째 printf문에서 출력하고자 하는 데이터는 'array1[0]+5'인데, 이것은 array1[0]이 가리키는 곳 'G'으로부터 5칸 뒤의 값을 가리킨다.
그러므로 array1[0]+5가 가리키는 위치는 'm'이 되고, %s를 이용해 문자열 스트림을 출력하면 'm'부터 끝까지인, 'morning'가 출력이 된다.

두 번째 printf문에서 '%c'는 문자 하나를 출력한다는 의미이다.
두 번째 printf문에서 출력하고자 하는 데이터는 'array1[1]+6'인데, array1[1]이 가리키는 곳 'C'으로부터 6칸 뒤의 값인 'u'를 가리킨다는 의미다. %c는 해당 위치 문자 하나만 출력하기 때문에 'u'만 출력된다.

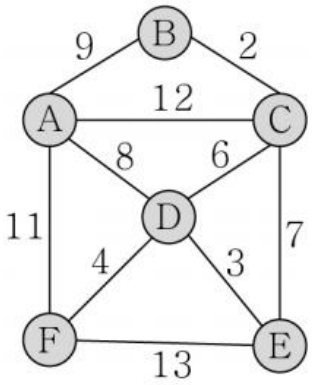
8. 메모리 크기가 200KB인 시스템에서 요구 페이징(demand paging)으로 가상 메모리(virtual memory)를 구현한다고 하자. 페이지 크기가 2KB이고 페이지 테이블(page table)의 각 항목이 3바이트라고 하면, 25KB 크기의 프로세스를 위한 최소 페이지 테이블의 크기는 어떻게 되는가?

① 25바이트
② 39바이트
③ 60바이트
④ 75바이트

답 ②

페이지 하나의 크기가 2KB이므로, 25KB 크기의 프로세스를 위해서는 최소 13개의 페이지가 필요하다.
페이지 테이블의 항목 1개가 3바이트라고 하였으므로, 13개의 항목의 3×13 = 39 바이트가 된다.

9. 다음 그래프에서 최소 비용의 신장 트리 값은 얼마인가?

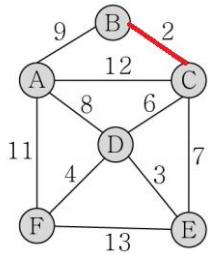


① 16 ② 20
③ 23 ④ 26

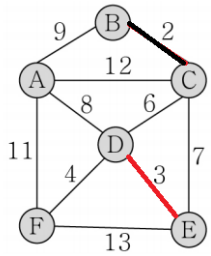
답 ③

9급 컴퓨터일반 시험에 출제된 최소 비용 신장트리 알고리즘은 Kruskal 알고리즘과 Prim 알고리즘 두 가지가 있는데, 문제에 언급되어 있지 않을 땐 거의 Kruskal 알고리즘으로 풀면 된다.
또한 Prim 알고리즘은 시작 위치가 지정되어 있는데, 문제에 시작점이 지정되어 있지 않으므로, Kruskal 알고리즘으로 풀라고 낸 문제이다.
Kruskal 최소 비용 신장 트리는 가중치 값이 작은 것부터 순차적으로 선택해나가는 방식이다. 단 두 노드가 이미 연결이 되어 있는 상태라면 무시하고 다음 가중치 값으로 넘어가며, 모든 노드가 다 연결될 때까지 진행한다.
노드가 n개 있다면, 선택하는 간선의 수는 n-1개이면 족하다.

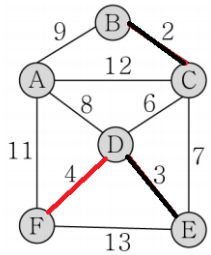
1. 먼저 가장 값이 작은 2(B-C)를 선택한다.



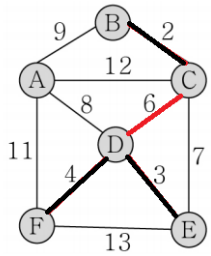
2. 그 다음으로 작은 3(D-E)를 선택한다.



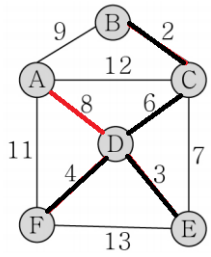
3. 그 다음으로 작은 4(D-F)를 선택한다.



4. 그 다음으로 작은 6(D-C)를 선택한다.



5. 그 다음으로 작은 수는 7인데, 7번 간선의 양 끝의 노드 C와 E는 이미 D를 통해 연결되어 있는 상태이기 때문에, 선택할 필요가 없다.
따라서, 그 다음 작은 수인 8(A-D)를 선택한다.



이렇게 하면 모든 노드가 선으로 다 연결이 되며, 이 때의 가중치 값을 다 더하면 2+3+4+6+8 = 23이 된다.

10. 라우팅 알고리즘은 라우터에 패킷이 도착했을 때 포워딩 테이블을 검색하고 패킷이 전달될 인터페이스를 결정하는 알고리즘이다. 다음 중 라우팅 알고리즘이 아닌 것은?

① RIP(Routing Information Protocol)
② OSPF(Open Shortest Path First)
③ CDMA(Code Division Multiple Access)
④ BGP(Border Gateway Protocol)

답 ③

③ **CDMA**(Code Division Multiple Access, 코드분할 다중접속)은 코드를 이용한 다중 접속 기술로, 하나의 통신망을 다수의 사용자가 분할 사용하는 방법 중 하나이다. 경로를 선택하고 라우팅 정보를 전송하는 라우팅 알고리즘과는 관계없다.

◈ IGP(Interior Gateway Protocol, 내부 게이트웨이 프로토콜)
▷ **RIP**(Routing Information Protocol)
- 거리벡터 알고리즘
- 소규모 또는 교육용 등 비교적 간단한 네트워크에 주로 사용
- 경로비용을 단지 홉 수(hop count)로만 판단
- 속도나 거리 지연 등을 고려하지 않아 최적의 경로 산정에 비효율적
- 라우팅 정보의 변화가 없더라도 매 30초 마다 자동으로 라우팅 정보를 브로드캐스팅하는데, 이는 트래픽에 부하가 줌

▷ **OSPF**(Open Shortest Path First)
- 링크상태 라우팅 프로토콜
- 자치시스템(AS) 내부의 라우터들끼리(IGP) 라우팅 정보를 교환
- 최단 경로를 선택하기 위해 Dijkstra의 SPF(Shortest Path First) 알고리즘을 사용
- 링크 상태의 변화시에만 라우팅 정보를 전송

▷ 이 외 IGRP, IS-IS 등

◈ EGP(Exterior Gateway Protocol, 외부 게이트웨이 프로토콜)
▷ **BGP**(Border Gateway Protocol)
- 프로토콜자치시스템(AS) 상호 간에 적용되는 라우팅 프로토콜
- 순환을 피할 수 있도록 목적지까지 가는 경로 정보를 제공
- 발전된 거리 벡터 라우팅 프로토콜 또는 경로 벡터 라우팅 프로토콜
- BGP는 주기적으로 정보를 갱신하지 않고, 단지 변화가 있을때만, 이웃 라우터에게 갱신 정보 전송
- 네트워크 변화가 전혀 없으면 주고받는 정보가 없게 되는데, 이를 위해 자신이 살아있음을 알리는 BGP 킵얼라이브 메세지(BGP Keepalive Message)를 60초 마다 교환

11. 암달(Amdahl)의 법칙은 컴퓨터 시스템의 일부를 개선할 때 전체적으로 얼마만큼의 최대 성능 향상을 기대할 수 있는지를 예측하는 데 사용된다. 만약 특정 응용프로그램의 75%가 멀티코어(Multicore)를 이용한 병렬 수행이 가능하고 나머지 25%는 코어의 수가 증가해도 순차 실행만 가능하다는 전제 하에, 컴퓨팅 코어(Core)의 수를 4개로 늘릴 때 기대할 수 있는 최대 성능 향상은 약 몇 배인가?

① 약 1.28배 ② 약 2.28배
③ 약 3.28배 ④ 약 4.28배

답 ②

(사실 이 문제는 암달의 법칙이니 하는 이론을 모르더라도 수학적인 사고를 통해 푸는 게 더 익숙하고 정확할 것이다.)
기존 시스템으로 100시간이 걸리는 일을 한다고 생각하자.
그 중 25시간분은 코어의 수에 상관없이 순차 실행만 가능한 일이고, 75시간분은 코어를 늘리면 병행 처리가 가능한 일이다.
병행 처리가 가능한 75시간 부분은 코어를 4배로 늘리면 시간을 1/4, 즉 18.75시간으로 줄일 수 있다.
그렇게 되면 기존 100시간이 걸리던 일을, 25 + 18.75 = 43.75 시간으로 줄일 수 있게 된다.
성능 향상은 100÷43.75 = 2.286배가 된다.

(암달의 공식 정의도 이 풀이 방식과 거의 동일하며, 다만 이걸 수식으로 표현한 것일 뿐이다.)

12. <보기> 중 우리가 흔히 인터넷을 통해 비용을 지불하거나 혹은 무료로 사용하는, 클라우드 저장 서버에 대한 분류로 옳은 것을 모두 고르면?

〈 보 기 〉
ㄱ. Public cloud
ㄴ. Private cloud
ㄷ. Software as a service(Saas)
ㄹ. Platform as a service(Paas)
ㅁ. Infrastructure as a service(Iaas)

① ㄱ, ㄷ ② ㄱ, ㅁ
③ ㄴ, ㄷ ④ ㄴ, ㄹ

답 ②

ㄱ. Public cloud(공용 클라우드)
서비스 제공업체(벤더)가 구축한 서버, 스토리지 등의 IT 인프라를 기업이나 외부 사용자들이 사용료를 내고 이용하는 방식

ㅁ. **IaaS**(Infrastructure as a service)
클라우드를 이용해, 서버 자원, IP, 네트워크, 저장 공간, 전력 등 컴퓨팅 인프라를 사용하는 데 필요한 자원을 서비스하는 방식이다. 인터넷 상에서 단순 데이터 저장 기능만 제공하는 클라우드 저장 서버가 여기에 해당한다.

<오답 체크> ㄴ. Private cloud(사설 클라우드)
기업들이 자체적으로 데이터센터 안에 클라우드 환경을 구축해 사용하는 방식

ㄷ. **Saas**(Software as a service)
모든 서비스가 클라우드에서 이뤄지는 서비스 방식으로, 별도의 소프트웨어 설치 없이, 클라우드 환경에서 소프트웨어를 이용할 수 있는 방식이다.
대표적으로 이메일 서비스, MS 오피스 365, 구글 문서 등이 있다.

ㄹ. **Paas**(Platform as a service)
서비스를 개발 할 수 있는 안정적인 플랫폼(platform)과 그 환경을 이용하는 응용 프로그램을 개발 할 수 있는 API를 제공하는 서비스 방식이다.

13. 16진수로 표현된 $B9E_{(16)}$를 2진수로 표현하면 다음 중 무엇인가?

① 1100 0101 1101 (2)
② 0101 0101 1001 (2)
③ 1011 1001 1110 (2)
④ 1110 0101 1101 (2)

답 ③

16 = 2^4 이므로, 16진수 한 자리는 2진수로 4자리로 정확히 바뀐다.
$B_{(16)}$는 10진수 11에 해당하고 8+2+1이므로, 2진수 1011이다.
$9_{(16)}$은 10진수 9에 해당하고 8+1이므로, 2진수 1001이다.
$E_{(16)}$은 10진수 14에 해당하고 8+4+2이므로, 2진수 1110이다.

14. 다음 프로그램의 구성 요소들 중 프로세스 내에서 생성한 스레드들 사이에 공유되지 않는 것을 모두 고르면?

> ㄱ. 레지스터(Register)
> ㄴ. 힙(Heap) 메모리
> ㄷ. 전역 변수(Global variables)
> ㄹ. 스택(Stack) 메모리

① ㄱ, ㄴ ② ㄱ, ㄹ
③ ㄴ, ㄷ ④ ㄷ, ㄹ

답 ②

스레드(thread)는 프로세스보다도 작은 실행 흐름의 최소 단위로, 경량 프로세스라고도 한다.
스레드는 그들이 속한 프로세스의 자원들과 메모리를 공유하지만, 스레드 **자신만의 스택(stack)과 레지스터(register)를 가지며** 독립된 제어 흐름을 가진다.

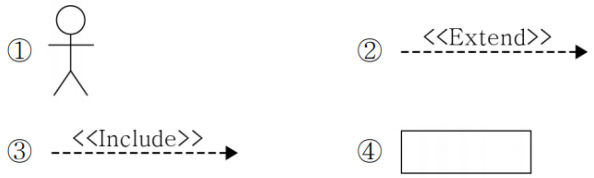
ㄱ. **레지스터**(register)
프로세서에 위치한 고속 메모리로 처리중인 중간 결과값 등 프로세서가 바로 사용할 수 있는 데이터를 일시적으로 저장하는 공간

ㄹ. **스택**(stack) 메모리
함수 호출 시 생성되는 지역 변수와 매개 변수가 저장되는 영역
함수 호출이 완료되면 사라짐

<오답 체크> ㄴ. **힙**(heap) 메모리
프로그래머가 관리하는 메모리 영역이다. 필요에 의해서 동적으로 할당·반납하는 메모리 공간

ㄷ. **전역 변수**(global varaiables)
모든 영역에서 접근할 수 있는 변수이므로 스레드들 사이에서도 당연히 공유된다.

15. 다음 중 Use case diagram에서 사용하는 기본 요소가 아닌 것은?



답 ④

④ Use case diagram(유스 케이스 다이어그램)에서 유스케이스는 직사각형이 아니라, 타원형이다.
Use case(유스케이스) 사용자 입장에서 바라본 시스템의 기능

<오답 체크> ① Actor(액터, 행위자) 시스템 기능을 사용하는 이용자

② Extend(확장) 유스케이스와 유스케이스 사이에 서비스 수행을 요청하는 관계
선택적으로 수행하는 관계이며, 옵션이라고 볼 수 있다.

③ Include(포함)도 마찬가지로 유스케이스가 다른 유스케이스의 서비스 수행을 요청하는 관계인데, 한 유스케이스가 자신의 서비스 수행 도중에 다른 유스케이스의 서비스 사용이 필요할 때 정의
Extend와 달리 Include 서비스는 반드시 사용되어야 하는 서비스이다.
예를 들어 예금 계좌이체를 수행하는데, 계좌 이체를 하기 위해서는 (내부적으로) 잔액 확인이 반드시 필요하다. 따라서 계좌 이체 서비스와 잔액 확인 서비스는 Include 관계이다.
계좌 이체를 완료한 후 추가로 이체할 것인지 물어보는데, 이 때 추가 이체는 할 수도 있고 안 할 수도 있기 때문에, Extend 관계에 해당한다.

16. 가상메모리(Virtual Memory)를 효과적으로 제공하기 위해 Core i7과 같은 프로세서 내부에 있는 장치는 무엇인가?

① TLB(Translation Lookaside Buffer)
② 캐시(Cache)
③ 페이지 테이블(Page Table)
④ 스왑 스페이스(Swap Space)

답 ①

① **TLB**(Translation Lookaside Buffer, 변환 색인 버퍼)
가상 메모리 주소를 물리적인 주소로 변환하는 속도를 높이기 위해 사용되는 캐시로, 최근에 일어난 가상 메모리 주소와 물리 주소의 변환 테이블을 저장하는 일종의 주소 변환 캐시이다.

<오답 체크> ② **캐시**(cache)는 데이터나 값을 미리 복사해 놓는 임시 저장 공간을 말한다.
CPU(중앙처리장치)와 RAM(주기억장치) 사이에서 속도 차이를 조절하기 위해 사용하는 CPU캐시와, 디스크에 입출력되는 데이터를 미리 읽어 불러오는 디스크 캐시 등이 있다.

③ 페이지 테이블(Page Table) 프로세스의 페이지 정보를 저장하고 있는 테이블

④ **스왑 스페이스**(Swap Space=Swap File=Swap Memory)
가상 메모리의 확장으로 사용되는 하드디스크 상의 한 공간이다.
새로운 데이터가 램에 들어오면, 가장 오래 전 사용되었던 데이터들은 새로운 데이터에 공간을 내어주고 하드디스크로 이동하는데, 이 공간을 스왑 스페이스(스왑 메모리)라고 한다.
가상 메모리와 같은 개념이라 생각하면 된다.

17. 다음 중 C 프로그래밍 언어의 식별자로 사용할 수 없는 것은?

① 3id
② My_ID
③ __yes
④ K

답 ①

① C언어에서 식별자는 숫자로 시작할 수 없다.

▷ 식별자(identifier)란 변수, 상수, 함수, 사용자 정의 타입 등을 정의할 때 사용하는 '이름'을 의미한다. 식별자를 정의할 땐 몇 가지의 제한 규칙이 있다.
1. 영문 대소문자, 숫자, 언더스코어(_)만 사용할 수 있다.
2. C언어에서 명령이나 특수한 용도를 위해 미리 정의해놓은 키워드는 식별자로 사용할 수 없다.
auto, break, case, char, const, continue, default, do, double, else, enum, extern, float, for, goto, if, inline, int, long, register, restrict, return, short, signed, sizeof, static, struct, switch, typedef, union
3. 숫자로 시작될 수 없으며, 반드시 영문자나 언더스코어(_)로 시작되어야 한다.

18. 3개의 page를 수용할 수 있는 메모리가 있으며, 현재 완전히 비어 있다. 어느 프로그램이 <보기>와 같이 page 번호를 요청했을 때, LRU(Least-Recently-Used)를 사용할 경우 몇 번의 page-fault가 발생하는가?

〈 보 기 〉
요청하는 번호순서 : 2 3 2 1 5 2 4 5

① 6번 ② 5번
③ 4번 ④ 3번

답 ②

▶ **LRU**(Least Recently Used)
가장 오랫동안 오랫동안 사용하지 않은 페이지를 교체

요청 페이지	2	3	2	1	5	2	4	5
P1	2	2	2	2	2	2	2	2
P2		3	3	3	5	5	5	5
P3				1	1	1	4	4
페이지 폴트	◎	◎		◎	◎		◎	

19. 소프트웨어 프로젝트 관리가 어려운 이유로 옳지 않은 것은?

① 소프트웨어는 형태가 없어 프로젝트 관리자는 프로젝트 진척사항을 분석하는 데 어려움이 있다.
② 소프트웨어 개발 프로세스는 조직에 따라 가변적이므로 관리에 어려움이 있다.
③ 컴퓨터와 통신에서의 빠른 기술적 변화로 인해 관리자의 경험이 새로운 프로젝트에 전달되지 않을 수 있다.
④ 대규모 소프트웨어 프로젝트는 일회성(one-off) 프로젝트가 전혀 없어서, 경험이 충분한 관리자가 문제를 예측할 수 없다.

답 ④

④ 대규모 소프트웨어 프로젝트는 종종 일회성(one-off) 프로젝트이다. 기존의 프로젝트와 방향성이 달라서, 경험이 충분한 관리자라도 문제를 예측하기 어렵다.
일회성 프로젝트가 전혀 없다면 기존의 프로젝트에서 활용할 수 있는 부분이 많기 때문에, 경험이 충분한 관리자가 문제를 예측하기 상대적으로 쉽다.

20. 다음 C 프로그램의 실행 결과는?

```
#include <stdio.h>
struct student
{
    char name[20];        // 이름
    int money;            // 돈
    struct student* link;  // 자기 참조 구조체 포인터 변수
};
int main(void)
{
    struct student stu1 = {"Kim", 90, NULL };
    struct student stu2 = {"Lee", 80, NULL };
    struct student stu3 = {"Goo", 60, NULL };
    stu1.link = &stu2;
    stu2.link = &stu3;
    printf("%s %d ₩n", stu1.link->link->name,
        stu1.link->money);
    return 0;
}
```

① Goo 80 ② Lee 60
③ Goo 60 ④ Lee 80

답 ①

stu1의 주소를 #010, stu2의 주소를 #020, stu3의 주소를 #030으로 가정하자.
stu1.link = &stu2; 는 stu2의 주소를 stu1 구조체의 link 위치에 저장한다는 의미이다.
따라서 stu1의 값은 { "Kim", 90, #020 } 으로 바뀐다.
그 다음 stu2.ink = &stu3; 를 통해
stu2의 값은 { "Lee", 80, #030 } 으로 바뀐다.

stu1{ "Kim, 90, #020 } -> stu2{ "Lee", 80, #030 } -> stu3{"Goo", 60, NULL }

printf문에서 %s는 문자열 출력을 의미하며,
stu1.link->link->name 는
stu1.link에 저장된 주소(#020)을 주소로 가지는 구조체(stu2)의 link에 저장된 주소(#030)을 주소로 가지는 구조체(stu3)의 name 값을 가리킨다.
따라서 stu3.name 에 해당하는 "Goo"가 된다.

printf문의 %d는 정수 출력을 의미하며,
stu1.link->money 는
stu1.link에 저장된 주소(#020)을 주소로 가지는 구조체(stu2)의 money에 저장된 값을 가리킨다.
따라서 stu2.money 에 해당하는 '80'이 된다.