2021년 국가직 9급 컴퓨터일반 풀이

by 호이호이꿀떡

정답 체크

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

문 1. 컴퓨팅 사고(Computational Thinking)에서 주어진 문제의 중요한 특징만으로 문제를 간결하게 재정의함으로써 문제 해결을 쉽게 하는 과정은?

① 분해
② 알고리즘
③ 추상화
④ 패턴 인식

◈ 컴퓨팅 사고(CT, computational thinking)
컴퓨터(사람이나 기계)가 효과적으로 수행할 수 있도록 문제를 정의하고 그에 대한 답을 기술하는 것이 포함된 사고 과정 일체.
추상화, 분해, 패턴 인식, 알고리즘으로 구성된다.

▶ 추상화(abstraction)
문제에서 중요하지 않은 부분을 제거하고 중요한 특징만으로 문제를 구성함으로써 문제 해결을 좀 더 쉽게 하는 과정
추상화로 공통의 특성을 추려 내어 만든 개념이 일반화(generalization)이다.
예) 전체 지도에서 지하철 정보만 모은 지하철 노선도

▶ 문제 분해(decomposition)
추상화한 (복잡한)문제를 해결하기 쉬운 작은 단위의 문제로 나누는 과정
예) 라면 끓이기를 물 끓이기, 면 삶기, 스프 넣기, 계란 풀기 등으로 나누는 것
이렇게 나눈 작은 문제들을 해결하고, 해결된 작은 문제를 결합하여 큰 문제를 해결하는 방식을 분해 정복(divide and conquer)이라고 한다.

▶ 패턴 인식(pattern recognition)
추상화 및 분해를 한 후 데이터를 특징별로 나누어 문제에 적용 가능한 패턴을 찾아내는 과정.
패턴을 찾을 수 잇다면 문제를 해결하기 매우 쉽다.
예) 평일보다 휴일에 식당에 손님이 많이 몰리는 특징을 통해서, 휴일에만 식당에 테이블을 추가하는 방식을 도입할 수 있다.

▶ 알고리즘(algorithm)
어떤 문제를 해결하기 위해 정해진 일련의 절차나 방법을 기술한 것
예) 요리 레시피 등

답 ③

문 2. 소프트웨어에 대한 설명으로 옳지 않은 것은?

① 하드웨어에 대응하는 개념으로 우리가 원하는 대로 컴퓨터를 작동하게 만드는 논리적인 바탕을 제공한다.
② 운영체제 등 컴퓨터 시스템을 가동시키는 데 사용되는 소프트웨어를 시스템 소프트웨어라 한다.
③ 문서 작성이나 게임 등 특정 분야의 업무를 처리하는 데 사용되는 소프트웨어를 응용 소프트웨어라 한다.
④ 고급 언어로 작성된 프로그램을 한꺼번에 번역한 후 실행하는 것이 인터프리터 방식이다.

④ **컴파일러**(compiler)에 대한 설명이다.
특정 프로그래밍 언어(고급 언어)로 작성된 원시 코드를 한번에 번역하여 실행하는 방식으로, 목적 프로그램으로 생성한다.
인터프리터(interpreter)는 목적 프로그램을 생성하지 않고 원시 코드를 한 줄씩 바로바로 번역하여 실행하는 프로그램이다.

◈ **시스템 소프트웨어**(System Software)
시스템 운영을 위한 여러 제어, 관리 역할을 하여 하드웨어를 작동하고 운영하는 가장 기본적인 소프트웨어를 말한다. 응용 소프트웨어를 실행하기 위한 플랫폼을 제공한다.
윈도우나 안드로이드, IOS 등의 운영체제는 물론 장치 드라이버, 어셈블러, 컴파일러, 로더, 링크 등을 포함한다.
◈ **응용 소프트웨어**(Application Software)
계산, 그래픽, 게임, 시뮬레이션 등 어느 특정한 기능을 수행하기 위해 사용자가 조작하는 소프트웨어를 말한다. 항해, 포격 제어, 급여 계산용 소프트웨어 등이 있다.
워드프로세서, 엑셀, 포토샵, 크롬 등 사용자가 설치해 사용하는 대부분의 프로그램이 응용 소프트웨어에 해당한다.

답 ④

문 3. 4GHz의 클록 속도를 갖는 CPU에서 CPI(Cycle per Instruction)가 4.0이고 총 10^{10}개의 명령어로 구성된 프로그램을 수행하려고 할 때, 이 프로그램의 실행 완료를 위해 필요한 시간은?

① 1초
② 10초
③ 100초
④ 1,000초

GHz의 G는 10의 9승, 즉 10억을 의미한다.
4GHz란 1초에 4 × G(기가) = 40억 개의 클록 사이클을 수행한다는 의미이다.

프로그램이 10^{10}개, 즉 100억 개의 명령어로 구성되어 있으며, 명령어 하나 당 4 클록 사이클을 가지고 있으므로, 이 프로그램의 전체 클록 사이클 수는 400억 개이다.

따라서 전체 프로그램의 실행 시간은
400억 ÷ 40억 = **10초**이다.

답 ②

문 4. −35를 2의 보수(2's Complement)로 변환하면?

① 11011100
② 11011101
③ 11101100
④ 11101101

35를 2진수로 변환하면, '00100011'
각 비트를 반전시키면 '11011100'이 되며, 이것은 1의 보수 값이 된다.
1의 보수 값에 1을 더하면 2의 보수 값이 되므로,
2의 보수 -35는 '**11011101**'이 된다.

답 ②

문 5. OSI 7계층에서 계층별로 사용하는 프로토콜의 데이터 단위는 다음 표와 같다. ㉠ ~ ㉢에 들어갈 내용을 바르게 연결한 것은?

계층	데이터 단위
트랜스포트(Transport) 계층	(㉠)
네트워크(Network) 계층	(㉡)
데이터링크(Datalink) 계층	(㉢)
물리(Physical) 계층	비트

	㉠	㉡	㉢
①	세그먼트	프레임	패킷
②	패킷	세그먼트	프레임
③	세그먼트	패킷	프레임
④	패킷	프레임	세그먼트

▶ 각 계층별 PDU(Process Data Unit)
1. 물리 계층 : 비트(bits), 1계층의 PDU를 비트로 보기도 하고, 그냥 하나의 신호로 보기도 한다.
2. 데이터 링크 계층 : 프레임(frames)
3. 네트워크 계층 : 패킷(packets)
4. 전송(트랜스포트) 계층 : 세그먼트(segments)
5. 세션 계층 ~ 7. 응용 계층 : 메시지(messages) 또는 데이터(data)

답 ③

문 6. 300개의 노드로 이진 트리를 생성하고자 할 때, 생성 가능한 이진 트리의 최대 높이와 최소 높이로 모두 옳은 것은? (단, 1개의 노드로 생성된 이진 트리의 높이는 1이다)

	최대 높이	최소 높이
①	299	8
②	299	9
③	300	8
④	300	9

■ **최대 높이**

높이를 최대로 만들기 위해서는, 트리의 한 쪽 방향으로만 노드를 채워나가면 된다. 그러면 모든 레벨(높이)에는 노드가 1개만 채워지고, 300개의 노드로 최대 300레벨의 이진 트리를 만들 수 있다.

이것을 사향 이진 트리(SBT, Skewed Binary Tree)라고 한다.

■ **최소 높이**

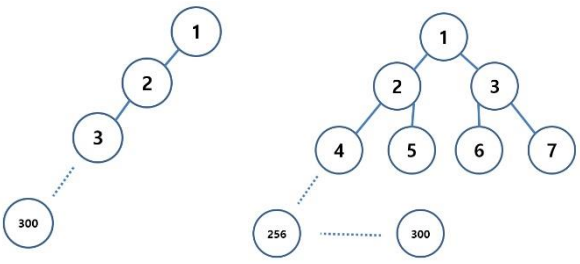
높이를 최소로 만들기 위해서는, 트리의 각 레벨을 노드로 가득 채워야 한다. 그러면 1레벨에는 노드 1개, 2레벨에는 노드 2개, 3레벨에는 노드 4개.... n 레벨에는 노드 2n-1개를 채울 수 있으며, 1~n 레벨까지의 노드 개수의 합은 2n-1개이다.

8레벨까지의 노드 개수의 합은 28-1 = 255개

9레벨까지의 노드 개수의 합은 29-1 = 511개이므로,

300개의 노드를 이용해 최소 9레벨의 이진 트리를 만들 수 있다.

이렇게 각 레벨 마다 노드를 가득 채우는 것을 전이진 트리(CBT, Complete Binary Tree)라고 한다.(단, 중간 노드를 비우지 않고 앞에서부터 채워나가야 한다)



답 ④

문 7. 아래와 같은 순서대로 회의실 사용 요청이 있을 때, 다음 중 가장 많은 회의실 사용 시간을 확보할 수 있는 스케줄링 방법은? (단, 회의실은 하나이고, 사용 요청은 (시작 시각, 종료 시각)으로 구성된다. 회의실에 특정 회의가 할당되면 이 회의 시간과 겹치는 회의 요청에 대해서는 회의실 배정을 할 수 없다)

(11 : 50, 12 : 30),	(9 : 00, 12 : 00),
(13 : 00, 14 : 30),	(14 : 40, 15 : 00),
(14 : 50, 16 : 00),	(15 : 40, 16 : 20),
(16 : 10, 18 : 00)	

① 시작 시각이 빠른 요청부터 회의실 사용이 가능하면 확정한다.

② 종료 시각이 빠른 요청부터 회의실 사용이 가능하면 확정한다.

③ 사용 요청 순서대로 회의실 사용이 가능하면 확정한다.

④ 회의 시간이 긴 요청부터 회의실 사용이 가능하면 확정한다.

① 시작 시각이 빠른 요청부터 배정
(9:00, 12:00) -> (13:00, 14:30) -> (14:40, 15:00) -> (15:40, 16:20)
3시간 + 1시간30분 + 20분 + 40분 = **5시간30분**

② 종료 시각이 빠른 요청부터 배정
(9:00, 12:00) -> (13:00, 14:30) -> (14:40, 15:00) -> (15:40, 16:20)
3시간 + 1시간30분 + 20분 + 40분 = **5시간30분**

③ 사용 요청 순서대로 배정
(11:50, 12:30) -> (13:00, 14:30) -> (14:40, 15:00) -> (15:40, 16:20)
40분 + 1시간30분 + 20분 + 40분 = 3시간10분

④ (9:00, 12:00) -> (16:10, 18:00) -> (13:00, 14:30) -> (14:50, 16:00)
3시간 + 1시간50분 + 1시간30분 + 1시간10분 = **7시간30분**

따라서 가장 많은 시간을 확보할 수 있는 방법은 **④번** 방법이다

답 ④

문 8. 제품 테이블에 대하여 SQL 명령을 실행한 결과가 다음과 같을 때,㉠과 ㉡에 들어갈 내용을 바르게 연결한 것은?

<제품 테이블>

제품ID	제품이름	단가	제조업체
P001	나사못	100	A
P010	망치	1,000	B
P011	드라이버	3,000	B
P020	망치	1,500	C
P021	장갑	800	C
P022	너트	200	C
P030	드라이버	4,000	D
P031	절연테이프	500	D

<제품 테이블>

```
SELECT 제조업체, MAX(단가) AS 최고단가
FROM 제품
GROUP BY ( ㉠ )
HAVING COUNT(*) > ( ㉡ ) ;
```

<제품 테이블>

제조업체	최고단가
B	3,000
C	1,500
D	4,000

㉠	㉡
① 제조업체	1
② 제조업체	2
③ 단가	1
④ 단가	2

㉠ 실행 결과를 보면, 제조업체 B, C, D의 제품들 중 가장 단가가 높은 제품의 가격을 추출하였다.
이는 전체 테이블에서 제조업체별로 그룹을 묶은 뒤, 해당 그룹 중 단가값이 가장 높은 것을 추출한 것이다.
-> 따라서, **GROUP BY 제조업체** 가 된다.

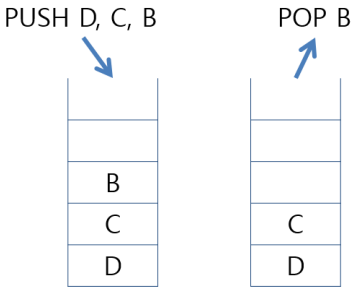
㉡ HAVING 절은 GROUP 절의 조건문이 된다.
실행 결과를 보면 제조업체 A가 제외되어 있으며, 전체 테이블에서 A의 행 개수는 1개이고, B, C, D의 행 개수는 2개 이상이다.
-> 따라서 HAVING 절은 행 개수가 2개 이상 또는 1개 초과인 조건문에 해당하고,
HAVING COUNT(*) > 1 이 정답이 된다.

답 ①

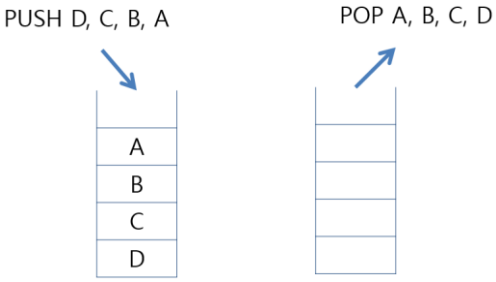
문 9. 스택의 입력으로 4개의 문자 D, C, B, A가 순서대로 들어올 때, 스택 연산 PUSH와 POP에 의해서 출력될 수 없는 결과는?

① ABCD
② BDCA
③ CDBA
④ DCBA

② B를 출력하기 위해선 먼저 D, C, B를 스택에 넣어야 하는데, 이 상태에서는 B를 출력하면 C가 가장 상단에 남아있어 D를 출력할 수 없다.



<오답 체크> ① 순서대로 D, C, B, A를 모두 스택에 집어넣은 뒤, 위에서부터 차례대로 A, B, C, D를 출력하면 된다.
PUSH D -> PUSH C -> PUSH B -> PUSH A
-> POP A -> POP B -> POP C -> POP A



③ 먼저 D, C를 스택에 넣은 뒤, C, D 순서로 출력하고,
B를 스택에 넣었다 바로 출력, A 역시 스택에 넣었다 바로 출력하면 된다.
PUSH D -> PUSH C -> POP C -> POP D
-> PUSH B -> POP B -> PUSH A -> POP A

④ D, C, B, A를 모두 스택에 넣자마자 바로 출력하면 된다.
PUSH D -> POP D -> PUSH C -> POP C
-> PUSH B -> POP B -> PUSH A -> POP A

답 ②

문 10. 임계구역에 대한 설명으로 옳은 것은?

① 임계구역에 진입하고자 하는 프로세스가 무한대기에 빠지지 않도록 하는 조건을 진행의 융통성(Progress Flexibility)이라 한다.
② 자원을 공유하는 프로세스들 사이에서 공유자원에 대해 동시에 접근하여 변경할 수 있는 프로그램 코드 부분을 임계영역(Critical Section)이라 한다.
③ 한 프로세스가 다른 프로세스의 진행을 방해하지 않도록 하는 조건을 한정 대기(Bounded Waiting)라 한다.
④ 한 프로세스가 임계구역에 들어가면 다른 프로세스는 임계구역에 들어갈 수 없도록 하는 조건을 상호 배제(Mutual Exclusion)라 한다.

④ 임계구역 해결 조건 중 하나인 '상호 배제'에 대한 설명이다.
<오답 체크> ① '한정 대기'에 대한 설명이다.
② 임계영역은 한 시점에는 하나의 프로세스만 공유자원에 접근하여 변경할 수 있어야 한다.
③ '진행의 융통성'에 대한 설명이다.

◈ 임계구역 해결 조건
● 상호 배제(mutual exclusion)
한 프로세스가 임계구역 내에 있을 때 다른 프로세스는 임계구역에 들어갈 수 없다.
● 한정 대기(bounded waiting)
임계구역에 진입하려는 프로세스가 무한대기에 빠지지 않아야 한다.
● 진행의 융통성(progress flexibility)
한 프로세스가 다른 프로세스의 진행을 방해하면 안 되며, 임계구역이 비어 있으면 프로세스는 항상 임계구역에 진입할 수 있어야 한다.

답 ④

문 11. 통합 테스팅 방법에 대한 설명으로 옳지 않은 것은?

① 연쇄식(Threads) 통합은 초기에 시스템 골격을 파악하기 어렵다.
② 빅뱅(Big-bang) 통합은 모든 모듈을 동시에 통합하여 테스팅한다.
③ 상향식(Bottom-up) 통합은 가장 하부 모듈부터 통합하여 테스팅한다.
④ 하향식(Top-down) 통합은 프로그램 제어 구조에서 상위 모듈부터 통합하는 것을 말한다.

◈ 통합 테스팅
단위 테스트가 끝난 모듈을 통합하는 과정에서 발생할 수 있는 오류를 찾는 테스트. 모듈들 사이에 인터페이스 오류는 없는지, 모듈이 올바르게 연계되어 동작하는지를 테스트한다.
▶ 빅뱅 접근법
전체 시스템을 한번에 테스트하는 방식.
장애가 발생한 위치를 파악하기 어렵다는 단점 존재
▶ 연쇄식 통합 테스트
시스템의 중요한 기능을 담당하는 모듈부터 통합하는 방식
초기에 시스템 골격을 보여주고 사용자의 의견을 받아 수정이 가능하다
▶ 점증적 접근법
연관된 모듈을 하나씩 추가해가면서 테스트하는 방식
상향식 접근법과 하향식 접근법이 있다.
▶ 상향식 접근법
하위 레벨에 있는 모듈들부터 시작해, 하나씩 상위 레벨 모듈을 추가해가면서 테스트하는 방식
장애가 발생한 위치를 파악하기 쉬운 반면, 전체 애플리케이션의 흐름을 테스트하기에는 시간이 걸린다는 단점이 있다.
▶ 하향식 접근법
상위 레벨에 있는 모듈들부터 시작해, 하나씩 하위 레벨 모듈을 추가해가면서 테스트하는 방식
장애가 발생한 위치를 파악하기 쉽지만, 낮은 레벨 모듈들을 충분히 테스트하기 어렵다.
▶ 샌드위치 테스트
상향식 접근법과 하향식 접근법을 결합한 방식
하위 프로젝트가 있는 대규모 프로젝트에 사용하는 방식이며, 병렬 테스트가 가능해서 시간이 절약된다. 반면에 많은 비용이 소모된다.

답 ①

문 12. 다음 중 파이썬 프로그래밍 언어에 대한 설명으로 옳은 것만을 모두 고르면?

> ㄱ. 변수 선언 시 변수명 앞에 데이터형을 지정해야 한다.
> ㄴ. 플랫폼에 독립적인 대화식 언어이다.
> ㄷ. 클래스를 정의하여 객체 인스턴스를 생성할 수 있다.

① ㄴ
② ㄱ, ㄷ
③ ㄴ, ㄷ
④ ㄱ, ㄴ, ㄷ

ㄴ. 파이썬은 플랫폼에 종속되지 않고 독립적이며, 윈도우, 리눅스, 맥 등 다양하게 활용이 가능하다.
ㄷ. 파이썬 역시 객체지향 언어로, 클래스를 정의하여 객체 인스턴스를 생성할 수 있다.
<오답 체크> ㄱ. 파이썬에서는 변수를 선언할 때 자료형을 지정하지 않는다.

● **파이썬(Python)의 특징**
스크립트 언어 사용, 인터프리터로 결과 바로 확인
변수 자료형 지정하지 않고 선언(동적 타이핑)
플랫폼에 독립적인 언어로, 윈도우, 리눅스, 맥 모두 호환
문법이 간결하고 쉬우며, 개발이 편하고 빠르다
높은 확장성 및 이식성
컴파일 언어에 비해 실행 속도 느림
GIL(Global Interpreter Lock)에 의해 쓰레드 병렬 실행이 제한됨

답 ③

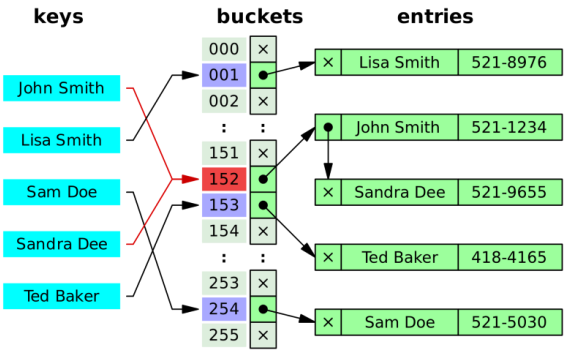
문 13. 해쉬(Hash)에 대한 설명으로 옳지 않은 것은?

① 연결리스트는 체이닝(Chaining) 구현에 적합하다.
② 충돌이 전혀 없다면 해쉬 탐색의 시간 복잡도는 O(1)이다.
③ 최악의 경우에도 이진 탐색보다 빠른 성능을 보인다.
④ 해쉬 함수는 임의의 길이의 데이터를 입력받을 수 있다.

▶ 해쉬(Hash) 방식이란, 값(데이터)을 저장, 검색할 때, 값을 직접 저장, 검색하지 않고, 키 값과 해쉬를 이용해 저장하는 방식이다.
먼저, 값(value)에 대응하는 키(key) 값을 지정해서 해시 테이블(Hash Table)을 구성한다. 해시 테이블에는 값과 키가 1:1 대응하여 저장되어 있으서, 해시 테이블에서 키 값을 찾으면 해당하는 실제 값을 바로 찾을 수 있다.
그리고 키를 해시 함수를 통해 해시값으로 변환하여 저장 위치를 계산한 뒤, 키를 슬롯에 저장한다. 예를 들어 키 A의 해시값이 3이면, 3번 슬롯에 A값을 저장하고, 키 B의 해시값이 5이면, 5번 슬롯에 B값을 저장하는 방식이다.

▶ 만일 키 C와 키 D가 똑같이 해시값이 2라면, C값과 D값 모두 2번에 저장해야 하는 상황이 발생하고, 이것을 해쉬 충돌(Hash Collision)이라고 한다.

① 해쉬 충돌을 해결하기 위한 방법 중 하나인 체이닝(Chaining)은 충돌이 발생한 값들을 연결 리스트로 연결하는 방법이다.



위 그림처럼, John Smith값과 Sandra Dee값이 152번 슬롯에서 충돌했을 경우, 152번 슬롯 자리에 John Smith값을 집어넣고 Sandra Dee값을 John Smith값에 리스트로 연결하는 것이다.

② 충돌이 전혀 없다면, 모든 키의 해시값이 다 다르고 모든 값은 다 해시값에 맞는 슬롯에 저장되어 있는 상태이다.
이 때는 키 값을 계산하여 해시값만 구하면 바로 저장된 값을 찾을 수 있어, 시간복잡도는 O(1)이 된다.

③ 반면에, 최악의 경우 모든 키의 해시값이 동일하여 모든 값이 동일한 슬롯에서 충돌이 발생하고, 그 값들을 모두 연결 리스트로 연결하는 상황을 가정해보자.
이 때는 키를 통해 해시값을 구해봤자 바로 저장된 값을 찾을 수 없으며, 하나하나씩 연결 리스트를 따라가 원하는 값을 찾아야 한다. 따라서 이 때 최악의 경우 시간복잡도는 O(n)이 된다.

<정답> 이진 탐색의 시간복잡도는 O(log2n)이고 최악의 경우 해쉬 방식은 시간복잡도가 O(n)이므로, 최악의 경우에는 이진 탐색이 더 빠른 성능을 보인다.

④ 해쉬 함수는 임의의 길이의 데이터를 입력받아 고정된 길이의 해쉬값을 출력한다.

답 ③

문 14. 프로세스의 메모리는 세그먼테이션에 의해 그 역할이 할당되어 있다. 표준 C언어로 작성된 프로그램이 컴파일 후 실행파일로 변환되어 메모리를 할당받았을 때, 이 프로그램에 할당된 세그먼트에 대한 설명으로 옳은 것은?

① 데이터 세그먼트는 모든 서브루틴의 지역변수와 서브루틴종료 후 돌아갈 명령어의 주소값을 저장한다.
② 스택은 현재 실행 중인 서브루틴의 매개변수와 프로그램의 전역변수를 저장한다.
③ 코드 세그먼트는 CPU가 실행할 명령어와 메인 서브루틴의 지역변수를 저장한다.
④ 힙(Heap)은 동적 메모리 할당을 위해 사용되는 공간이고, 주소값이 커지는 방향으로 증가한다.

④ 힙 세그먼트는 메모리의 낮은 주소에서 높은 주소의 방향(주소값이 커지는 방향)으로 할당되며, 스택 세그먼트는 메모리의 높은 주소에서 낮은 주소의 방향(주소값이 작아지는 방향)으로 할당된다.

<오답 체크> ① 지역변수는 스택 세그먼트에 저장되고, 서브루틴 종료 후 돌아갈 명령어 주소는 링크 레지스터(LR, link register)에 저장된다.
② 매개변수는 스택 세그먼트에 저장되나, 전역변수는 데이터 세그먼트 또는 BSS 세그먼트에 저장된다.
③ 코드 세그먼트에는 프로그램 실행 코드가 저장된다.

스택(Stack) 세그먼트
힙(Heap) 세그먼트
BSS 세그먼트
데이터(Data) 세그먼트
텍스트(Text) 세그먼트 = 코드(Code) 세그먼트

▶ 스택 세그먼트
프로그램이 실행되는 동안 생성되는 데이터(**지역변수, 매개변수 등**)을 저장하는 영역이다.
해당 변수들은 함수 호출 시 생성되고 호출된 함수에서만 사용이 가능하며, 함수가 종료되면 이 변수들도 소멸된다.
쓰기 가능, 크기 가변

▶ 힙 세그먼트
필요에 의해 **동적으로 메모리를 할당**할 때 사용하는 영역이다.
쓰기 가능, 크기 가변

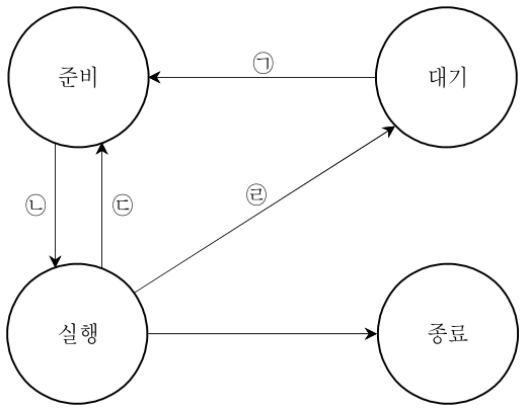
▶ BSS 세그먼트
초기화되지 않은 전역 변수와 정적(static) 변수가 저장되는 영역이다.
데이터 세그먼트는 초기에 사용할 메모리를 미리 확보하는 반면, BSS 세그먼트는 어느 정도의 메모리를 확보할지 정보만 미리 할당해놓고 있다가, 나중에 사용할 때 초기화한다.

▶ 데이터 세그먼트
(초기화된) 전역변수, 정적 변수, 일반상수, 문자열 등이 저장되는 영역이다. 해당 데이터들은 어느 함수에서나 호출이 가능하며, 프로그램이 시작될 때 생성되며, 프로그램이 종료될 때 소멸된다.
쓰기 가능, 크기 고정

▶ 텍스트(코드) 세그먼트
소스파일의 코드가 저장되는 영역이다. 코드가 담겨있기 때문에 쓰기 금지, 크기 고정

답 ④

문 15. 다음은 프로세스 상태 전이도이다. 각 상태 전이에 대한 예로 적절하지 않은 것은?



① ㉠ - 프로세스에 자신이 기다리고 있던 이벤트가 발생하였다.
② ㉡ - 실행할 프로세스를 선택할 때가 되면, 운영체제는 프로세스들 중 하나를 선택한다.
③ ㉢ - 실행 중인 프로세스가 자신에게 할당된 처리기의 시간을 모두 사용하였다.
④ ㉣ - 실행 중인 프로세스가 작업을 완료하거나 실행이 중단되었다.

④ ㉣ **블록(block)** : 실행(running) -> 대기(waiting)
즉시 자원을 할당받지 못해 자원을 할당받을 때까지 기다리게 되는 등, 당장 작업을 진행할 수 없어 다른 프로세스에 자리를 양보하고 대기 상태로 전환되는 것을 말한다.

<오답 체크> ① ㉠ **깨움(wakeup)** : 대기(waiting) -> 준비(ready)
② ㉡ **디스패치(dispatch)** : 준비(ready) -> 실행(running)
③ ㉢ **시간제한(타임아웃, timeout)** : 실행(running) -> 준비(ready)

답 ④

문 16. -30.25×2^{-8}의 값을 갖는 IEEE 754 단정도(Single Precision) 부동소수점(Floating-point) 수를 16진수로 변환하면?

① 5DF30000
② 9ED40000
③ BDF20000
④ C8F40000

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.

a) 10진수 -30.25를 2진수로 바꾸면 -11110.01이며,
이는 -1.111001×2^4 이다.
따라서 $-30.25 \times 2^{-8} = -1.111001 \times 2^4 \times 2^{-8}$
$= \mathbf{-1.111001 \times 2^{-4}}$

b) 음수(-)이기 때문에 부호는 **1**
c) 지수부(F)는 -4이며, 127 바이어스법을 따르기 때문에 127을 더하면 123, 즉 **01111011** 이다.
d) 가수부는 소수점 **11100100000000000000000** 이다.

따라서 값은 1 / 01111011 / 11100100000000000000000 이고,
이것을 16진수로 변환하면, 'BDF20000' 이 된다.

1011	1101	1111	0010	0000	0000	0000	0000
B	D	F	2	0	0	0	0

답 ③

문 17. 다음은 어느 학생이 C 언어로 작성한 학점 계산 프로그램이다. 출력 결과는?

```
#include <stdio.h>
int main()
{
    int score = 85;
    char grade;
    if (score >= 90) grade='A';
    if (score >= 80) grade='B';
    if (score >= 70) grade='C';
    if (score < 70) grade='F';
    printf("학점 : %c₩n", grade);
    return 0;
}
```

① 학점 : A
② 학점 : B
③ 학점 : C
④ 학점 : F

score는 85점이 들어가고,
첫 번째 if문에서 score는 90 이상이 아니므로 grade='A'는 실행되지 않는다.
두 번째 if문에서 score는 80 이상이므로, grade에는 B가 들어간다.
세 번째 if문에서 역시 마찬가지로 score는 70 이상이므로, grade에는 C가 들어간다.
네 번째 if문에서 score는 70보다 작지 않으므로, grade='F'는 실행되지 않는다.
▶ 따라서 최종적으로 **grade에는 C 값**이 들어있다.

● 보기에서 else if 문이 아니고 모두 if 문으로 되어 있기 때문에, 모든 if문을 차근히 전부 한번씩 실행한다.
원래 의도대로 만들기 위해선 2, 3, 4번째 if문을 else if문으로 변경해야 한다.

답 ③

문 18. 파이프라인 해저드(Pipeline Hazard)에 대한 다음 설명에서 ㉠과 ㉡에 들어갈 내용을 바르게 연결한 것은?

○ 하드웨어 자원의 부족 때문에 명령어를 적절한 클록 사이클에 실행할 수 있도록 지원하지 못할 때 (㉠) 해저드가 발생한다.
○ 실행할 명령어를 적절한 클록 사이클에 가져오지 못할 때 (㉡) 해저드가 발생한다.

	㉠	㉡
①	구조적	제어
②	구조적	데이터
③	데이터	구조적
④	데이터	제어

✠ 파이프라인 해저드(Pipeline Hazard)
▶ 구조적 해저드(Structural Hazard)
H/W의 자원 충돌로 인한 해저드
하드웨어가 여러 개의 명령어가 동시에 접근하는 것을 지원하기 않기 때문에 발생하며, 각각의 다른 명령어가 동일한 메모리, 동일한 레지스터에 동시에 접근하려 할 때 발생하는 해저드이다.
▶ 데이터 해저드(Data Hazard)
명령어의 선후 관계로 인한 해저드
명령어의 값이 현재 파이프라인 이전 명령어의 값에 종속되어 있을 때 발생하는 해저드이다.
파이프라인 구조에서 두 개 이상의 명령어 A, B를 동시에 수행할 때, B 명령어에서 처리할 데이터가 A 명령어가 수행 완료한 값이 필요할 경우, B는 A가 수행 완료될 때까지 대기해야 한다.
▶ 제어 해저드(Control Hazard)
분기 예측 실패로 인한 해저드
jump, branch 등 분기 명령어로 인해 기존에 실행 중이던 명령어들을 건너뛰어 불필요해지거나, 실행하지 않았어야 할 명령어들로 인해 데이터 값이 변경되는 생기는 해저드이다.

답 ①

문 19. 합성곱 신경망(CNN, Convolutional Neural Network) 처리 시 다음과 같은 입력과 필터가 주어졌을 때, 합성곱에 의해 생성된 특징 맵(Feature Map)의 ㉠에 들어갈 값은?

1	1	0	0
1	1	1	1
0	1	0	0
1	1	0	1

입력

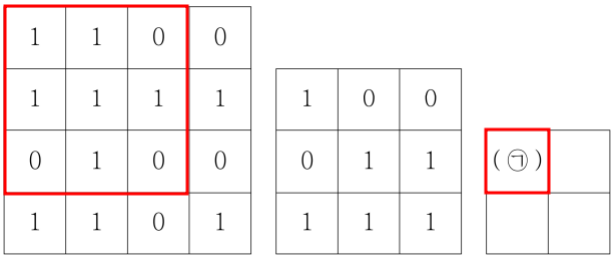
1	0	0
0	1	1
1	1	1

필터

(㉠)	

특징 맵

① 3
② 4
③ 5
④ 6

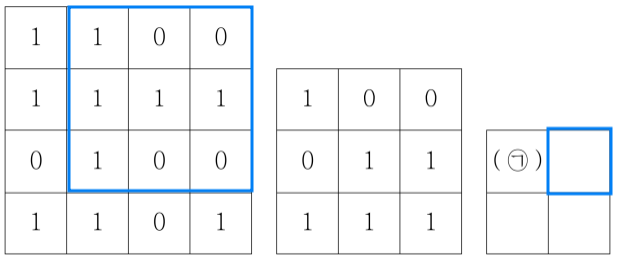


입력 맵 중 왼쪽 위 9개(빨간색)와 필터를 계산하여 특징 맵의 왼쪽 1번째 값을 계산한다.

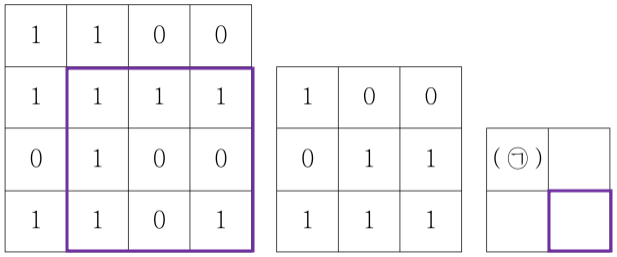
합성곱의 계산은 입력과 필터의 같은 자리 값들을 곱하여 계산하며,

(1×1)+(1×0)+(0×0) + (1×0)+(1×1)+(1×1)
+ (0×1)+(1×1)+(0×1) = 4, 따라서 **㉠에는 4**가 들어간다.

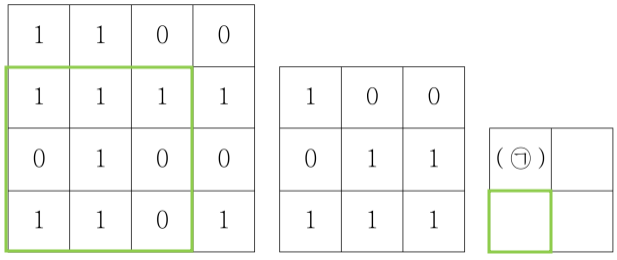
추가)



(1×1)+(0×0)+(0×0) + (1×0)+(1×1)+(1×1)
+ (1×1)+(0×1)+(0×1) = 4



(1×1)+(1×0)+(1×0) + (1×0)+(0×1)+(0×1)
+ (1×1)+(0×1)+(1×1) = 3



(1×1)+(1×0)+(1×0) + (0×0)+(1×1)+(0×1)
+ (1×1)+(1×1)+(0×1) = 4

따라서 최종적으로 특징 맵(출력)은 다음과 같이 된다.

4	4
3	4

답 ②

문 20. 해밍코드에 대한 패리티 비트 생성 규칙과 인코딩 예가 다음과 같다. 이에 대한 설명으로 옳은 것은?

<패리티 비트 생성 규칙>

원본 데이터	d4	d3	d2	d1			
인코딩된 데이터	d4	d3	d2	p4	d1	p2	p1

p1 = (d1 + d2 + d4) mod 2
p2 = (d1 + d3 + d4) mod 2
p4 = (d2 + d3 + d4) mod 2

<인코딩 예>

원본 데이터	0	0	1	1			
인코딩된 데이터	0	0	1	1	1	1	0

① 이 방법은 홀수 패리티를 사용하고 있다.
② 원본 데이터가 0100이면 0101110으로 인코딩된다.
③ 패리티 비트에 오류가 발생하면 복구는 불가능하다.
④ 수신측이 0010001을 수신하면 한 개의 비트 오류를 수정한 후 최종적으로 0010으로 복호한다.

	d4	d3	d2	p4	d1	p2	p1
수신 데이터	0	0	1	0	0	0	1
p1 검증 (맞음)	0		1		0		1
p2 검증 (맞음)	0	0			0	0	
p4 검증 (오류)	0	0	1	0			
오류 수정 후 데이터	**0**	**0**	**1**	**1**	**0**	**0**	**1**

1) p1 = (d1 + d2 + d4) mod 2 식이 성립하므로,
p1, d1, d2, d4는 오류가 없다고 볼 수 있다.
2) p2 = (d1 + d2 + d4) mod 2 식이 성립하므로,
p2, d1, d3, d4는 오류가 없다고 볼 수 있다.
3) p4 = (d2 + d3 + d4) mod 2 식이 성립하지 않으므로,
p4, d2, d3, d4 중 오류라고 볼 수 있으며,
위 1), 2)번을 통해 d2, d3, d4는 오류가 아니라고 했으므로,
오류가 발생한 비트는 **p4**이다.
최종적으로 복호화한 값은 d4, d3, d2, d1이므로 **0010**이 된다.

(추가) 해밍코드에서 오류가 발생한 자리를 찾는 정석 풀이
p4 p2 p1 순으로, 오류가 발생하면 1, 오류가 없으면 0을 집어넣어 2진수를 만든다.
p4은 오류가 발생했으므로 1, p2와 p1은 오류가 없으므로 0을 집어넣으면, $100_{(2)}$의 2진수가 나오며, $100_{(2)} = 4_{(10)}$
따라서, 4번째 비트인 p4가 오류 발생 비트이다.
(p1이 첫번째 비트임)

<오답 체크> ① 패리티 비트 p1, p2, p4를 검증하는 데, 1의 개수가 짝수여야 오류가 없는 정상값으로 판단한다. 따라서 **짝수 패리티**를 사용하고 있다.

② 원본 데이터가 0100일 때, **0101010**으로 인코딩된다.

	d4	d3	d2	p4	d1	p2	p1
원본 데이터	0	1	0		0		
p1 계산	0		0		0		0
p2 계산	0	1			0	1	
p4 계산	0	1	0	1			
인코딩 후 데이터	**0**	**1**	**0**	**1**	**0**	**1**	**0**

③ 패리티 비트 부분에 오류가 발생한 경우도 복구할 수 있다.
(위 ④번 풀이에서 패리티 비트 p4에 발생한 오류를 복구하였음)

답 ④