[소프트웨어 개발>데이터 입출력 구현>논리 데이터저장소 확인]

[이전 기출 문제]
1. 다음과 같은 전위식(prefix)을 후위식(postfix)으로 올바르게 표현한 것은?

- / * 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 -

[이전 기출 문제]
2. 다음의 수식을 후위 순회(postorder traversal)한 결과는?

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 / * * +

[이전 기출 문제]
3. 다음의 산술식을 “Postfix” 표기로 옳게 나타낸 것은?

X=A+(B+C/D)*E-F

① X=A+B+C/D*E-F　② XABCD/+E*+F-=
③ =X-+A*+B/CDEF　④ XABCDEF=++/*-

[이전 기출 문제]
4. 중위 표기법(infix)의 수식 (A+B)*C+(D+E)을 후위 표기법(postfix)으로 옳게 표기한 것은?

① AB+CDE*++　② AB+C*DE++
③ +AB*C+DE+　④ +*+ABC+DE

[이전 기출 문제]
5. 다음과 같은 후위식(prefix)을 중위식(infix)으로 올바르게 표현한 것은?

A B C - / D E F + * +

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

[소프트웨어 개발>애플리케이션 테스트 관리>애플리케이션 테스트 케이스 설계]

[기출 예상 문제]

1. 다음은 무엇에 대한 설명인가?

개발자의 장소에서 사용자가 개발자 앞에서 행해지며, 오류와 사용상의 문제점을 사용자와 개발자가 함께 확인하면서 검사하는 기법

① 알파 테스트 ② 베타 테스트
③ 통합 테스트 ④ 단위 테스트

[기출 예상 문제]

2. 다음 중 베타 테스트에 대한 설명으로 가장 적절한 것은?

① 구현된 단위 모듈의 기능 수행 여부를 판정하고, 내부에 존재하는 논리적 오류를 검출한다.
② 단위, 통합 테스트 후 전체 시스템이 정상적으로 작동하는지 판정하는 기능을 점검한다.
③ 다수의 사용자에게 제한되지 않은 환경에서 프로그램을 사용하게 하고 오류가 발견되면 개발자에게 통보한다.
④ 모듈 간의 인터페이스 연계를 검증하고 오류를 확인한다.

[기출 예상 문제]

3. 다음은 무엇에 대한 설명인가?

프로그램의 입력 조건에 타당한 입력 자료와 타당하지 않은 입력 자료의 개수를 균등하게 하여 테스트 케이스를 정하고, 해당 입력 자료에 맞는 결과가 출력되는지 확인하는 기법

① 동등분할 기법 ② 경계값분석 기법
③ 명세기반 기법 ④ 경험기반 기법

[기출 예상 문제]

4. 다음 중 원인 효과 그래프 기법에 대한 설명으로 옳은 것은?

① 탐색적 테스트, 즉흥적 테스트, 분류 트리 기법 등이 있다.
② 프로그램 명세서를 기반으로 테스트케이스를 선정하여 테스트하는 기법이다.
③ 입력 값과 출력 값을 트리 형태로 만들어서 특정 입력 값에 따라서 결정되는 출력 값이 정해지도록 만든 테스트 케이스 작성 기법이다.
④ 입력 값을 원인으로, 효과를 출력 값으로 정하고 이에 따른 원인 결과 그래프를 만들어서 테스트 케이스를 작성하는 기법이다.

[소프트웨어 개발>애플리케이션 테스트 관리>애플리케이션 성능 개선]

[기출 예상 문제]

1. 다음 중 알고리즘 조건에 대한 설명으로 가장 거리가 먼 것은?

① 입력: 알고리즘은 0 또는 그 이상의 외부에서 제공된 자료가 존재한다.
② 출력: 알고리즘은 0 또는 1개 이상의 결과를 가진다.
③ 명확성: 알고리즘의 각 단계는 명확하여 애매함이 없어야 한다.
④ 유한성: 알고리즘은 단계들을 유한한 횟수로 거친 후 문제를 해결하고 종료해야 한다.

[기출 예상 문제]

2. 다음 중 알고리즘 기법이 아닌 것은?

① 동등분할 기법 ② 분할 정복법
③ 동적 계획법 ④ 백트래킹

[기출 예상 문제]

3. 다음 빈 칸에 들어갈 용어로 가장 적절한 것은?

> 시간 복잡도는 프로그램을 실행시켜 완료하는데 걸리는 시간이다. 알고리즘의 소요 시간을 정확히 평가할 수는 없으므로, 자료의 수 n이 증가할 때 시간이 증가하는 대략적인 패턴을 시간 복잡도라는 이름으로 나타내게 된다. 이를 ( )(으)로 주로 나타낸다.

① 의사코드(pseudo-code)
② Big-O 표기법
③ 순서도(flowchart)
④ 버킷(Bucket)

[기출 예상 문제]

4. 다음은 무엇에 대한 설명인가?

> - 모든 경우의 수를 전부 고려하는 알고리즘. 상태공간을 트리(tree)로 나타낼 수 있을 때 적합한 방식이다. 일종의 트리 탐색 알고리즘이라고 봐도 된다.
> - 방식에 따라 깊이 우선탐색(Depth First Search, DFS)과 너비우선탐색(Breadth First Search, BFS) 등이 있다.

① 백트래킹(Backtracking)
② 그리디 알고리즘(Greedy Algorithm)
③ 동적 계획법(Dynamic Programming)
④ 분할 정복법(Divide and Conquer)

[소프트웨어 개발>애플리케이션 테스트 관리>애플리케이션 성능 개선]

[기출 예상 문제]

5. [100, 70, 90, 80, 90]을 선택 정렬을 사용하여 오름차순으로 정렬 할 경우, 다음 (ㄱ)~(ㄴ)에 들어갈 데이터로 가장 적절한 것은?

Pass 1: [70, 100, 90, 80, 90]
Pass 2: [70, 80, 100, 90, 90]
Pass 3: (ㄱ)
Pass 4: (ㄴ)

	(ㄱ)	(ㄴ)
①	[70,80,100,90,90]	[70,80,90,100,90]
②	[70,80,90,100,90]	[70,80,90,100,90]
③	[70,80,90,90,100]	[70,80,90,90,100]
④	[70,80,90,100,90]	[70,80,90,90,100]

[기출 예상 문제]

6. 알고리즘 정렬 기법 중 두 개씩 묶어서 정렬하는 기법은 무엇인가?

① 선택 정렬 ② 버블 정렬
③ 삽입 정렬 ④ 2-Way 병합 정렬

[기출 예상 문제]

7. [100, 70, 90, 80, 90]의 버블 정렬을 사용하여 오름차순으로 정렬 할 경우, 다음 (ㄱ)에 들어갈 데이터로 가장 적절한 것은?

Pass 1: [70, 100, 90, 80, 90]
Pass 2: (ㄱ)
Pass 3: [70, 80, 90, 90, 100]
Pass 4: [70, 80, 90, 90, 100]

① [70, 80, 90, 90, 100]
② [70, 90, 80, 90, 100]
③ [70, 80, 100, 90, 90]
④ [70, 100, 90, 90, 80]

[기출 예상 문제]

8. 다음은 무엇에 대한 설명인가?

시스템 검사의 종류 중 통합 시스템의 맥락에서 소프트웨어의 실시간 성능을 검사하며, 모든 단계에서 수행된다.

① 성능 검사 ② 모듈 검사
③ 시스템 검사 ④ 기능 검사

[정답] [소프트웨어 개발>추가문제]

1. 논리 데이터저장소 확인

1	2	3	4	5														
1	3	2	2	1														

2. 애플리케이션 테스트 케이스 설계

1	2	3	4															
1	3	1	4															

3. 애플리케이션 성능 개선

1	2	3	4	5	6	7	8											
2	1	2	1	4	4	1	1											

[정답 및 해설] [소프트웨어 개발>데이터 입출력 구현>논리 데이터저장소 확인]

논리 데이터저장소 확인 1. ①

전위식(연산자-피연산자-피연산자) → 후위식(피연산자-피연산자-연산자)
1. +BC → BC+
2. *ABC+ → ABC+*
3. /ABC+*D → ABC+*D/
4. -ABC+*D/E → ABC+*D/E-

논리 데이터저장소 확인 2. ③

왼쪽에서 오른쪽으로 연산자를 이동한다.
1. A / B * C * D + E
2. A B / * C * D + E
3. A B / C * * D + E
4. A B / C * D * + E
5. A B / C * D * E +

논리 데이터저장소 확인 3. ②

1. 괄호 먼저 제거: (B+C/D) → BCD/+
2. BCD/+*E → BCD/+E*
3. A+BCD/+E* → ABCD/+E*+
4. ABCD/+E*+-F → ABCD/+E*+F-
5. 등호는 항상 마지막에 처리: X=ABCD/+E*+F- → XABCD/+E*+F-=

논리 데이터저장소 확인 4. ②

후위 표기 방식은 중위 표기 방식으로 된 수식에서 연산자를 해당 피연산자 2개의 뒤(오른쪽)으로 이동한 것이다.
1. 인접한 피연산자 2개와 연산자를 괄호로 묶는다.
(((A + B) * C) + (D + E))
2. 연산자를 해당 괄호의 뒤로 옮긴다.
(((A B) + C) * (D E) +) +
3. 필요 없는 괄호 제거한다.
AB+C*DE++

논리 데이터저장소 확인 5. ①

1. 피연산자, 피연산자, 연산자 구조를 찾아서 괄호로 묶는다.
((A (B C -) /) (D (E F +) *) +)
2. (B C -) → (B - C)
3. (A (B C -) /) → (A / (B - C))
4. (E F +) → (E + F)
5. (D (E + F) *) → (D * (E + F))
6. ((A / (B - C)) (D * (E + F)) +)
→ ((A / (B - C)) + (D * (E + F)))
7. 필요 없는 괄호를 없앤다.
A/B-C+D*E+F

[정답 및 해설] [소프트웨어 개발>애플리케이션 테스트 관리>애플리케이션 테스트 케이스 설계]

애플리케이션 테스트 케이스 설계 1. ①

알파 테스트에 대한 설명이다.

애플리케이션 테스트 케이스 설계 2. ③

① 단위 테스트에 대한 설명이다.
② 시스템 테스트에 대한 설명이다.
④ 통합 테스트에 대한 설명이다.

애플리케이션 테스트 케이스 설계 3. ①

동등분할 기법(Equivalent Analysis)에 대한 설명이다.

애플리케이션 테스트 케이스 설계 4. ④

① 경험기반 기법에 대한 설명이다.
② 명세기반 기법에 대한 설명이다.
③ 결정트리 기법에 대한 설명이다.

[정답 및 해설] [소프트웨어 개발>애플리케이션 테스트 관리>애플리케이션 성능 개선]

애플리케이션 성능 개선 1. ②

② 출력: 알고리즘은 최소 1개 이상의 결과를 가진다.

애플리케이션 성능 개선 2. ①

* 알고리즘 기법
- 분할 정복법(Divide and Conquer)
- 동적 계획법(Dynamic Programming: DP)
- 그리디 알고리즘(Greedy Algorithm)
- 백트래킹(Backtracking)

애플리케이션 성능 개선 3. ②

알고리즘의 효율성 평가 시 Big-O 표기법으로 주로 나타낸다.

애플리케이션 성능 개선 4. ①

백트래킹(Backtracking)에 대한 설명이다.

애플리케이션 성능 개선 5. ④

- Pass3 일 때: [70, 80, 100, 90, 90]
  교환: [70, 80, 90, 100, 90]
  변화 없음: [70, 80, 90, 100, 90]
- Pass4 일 때: [70, 80, 90, 100, 90]
  교환: [70, 80, 90, 90, 100]

애플리케이션 성능 개선 6. ④

2-Way 병합 정렬에 대한 설명이다.

애플리케이션 성능 개선 7. ①

Pass 2 일 때: [70, 90, 80, 90, 100]
변화 없음: [70, 90, 80, 90, 100]
교 환: [70, 80, 90, 90, 100]
변화 없음: [70, 80, 90, 90, 100]

애플리케이션 성능 개선 8. ①

성능 검사에 대한 설명이다.