

Logical level은 도메인과 연산들을 추상적인 형태로 보여주는 것이다.	응답자 18 명	75 %	<div><div></div></div> ✓	33 % 정답
Application level은 User level과 동급으로 데이터 아이템의 구조를 서술하고, 연산들의 구체적인 구현에 대해 표현하는 것이다.	응답자 8 명	33 %	<div><div></div></div>	
Implementation level은 실생활의 데이터를 모델링하여 표현하는 것이다.	응답자 8 명	33 %	<div><div></div></div>	
도서관을 구현하려 할 때 Application level의 예로 책의 데이터를 표현하기 위한 자료구조와 그에 대한 연산 구현을 들 수 있다.	응답자 8 명	33 %	<div><div></div></div>	

시도 : 24 / 24

struct와 class 두 데이터 타입에 대한 다음 설명 중 맞는 것을 모두 고르시오.

동일한 class의 멤버함수는 메모리 상에 객체 수와 상관 없이 오직 한 개만 존재한다.	응답자 10 명	42 %	<div><div></div></div> ✓	4 % 정답
class의 멤버변수들은 모두 별개의 메모리 공간을 가진다.	응답자 22 명	92 %	<div><div></div></div>	
struct와 class는 서로 다르며 for while로 변환하여 표현하듯이 변환하여 구현하는 게 불가능하다.	응답자 4 명	17 %	<div><div></div></div>	
composite data type 분류에 따르면 두 데이터 타입은 structured 에 속한다.	응답자 8 명	33 %	<div><div></div></div>	

시도 : 24 / 24

1차원 배열의 구현에 대한 실제 내부 데이터 구조에 대한 다음 설명 중 맞는 것을 모두 고르시오.

1차원 배열의 시작 0번 인덱스 원소의 주소만 알면 원하는 index의 원소에 모두 랜덤 액세스가 가능하다.	응답자 17 명	71 %	<div><div></div></div> ✓	46 % 정답
index는 정수형 데이터 타입이어야 한다.	응답자 21 명	88 %	<div><div></div></div> ✓	
인덱스 메모리 주소 계산은 배열의 시작 주소 + index * 한 원소의 데이터 크기로 계산할 수 있다.	응답자 22 명	92 %	<div><div></div></div> ✓	
저장되는 원소는 동일한 데이터 타입을 가져야 한다.	응답자 18 명	75 %	<div><div></div></div> ✓	

class에 대한 다음 설명 중 맞는 것을 모두 고르시오.

class는 멤버 변수들과 그에 대해 처리하는 연산들인 멤버 함수들로 구성한다.	응답자 24 명	100 %	<div><div></div></div> ✓	0 % 정답
객체의 모든 멤버 변수들은 서로 별개의 메모리 공간에 존재한다.	응답자 18 명	75 %	<div><div></div></div>	
상속된 두 class간 관계는 'is a' 관계로 표현할 수 있다.	응답자 14 명	58 %	<div><div></div></div> ✓	
'has a' 관계는 포함 관계를 표현하는 것으로 class 내부에 다른 class의 객체를 멤버로 가지는 경우이다.	응답자 13 명	54 %	<div><div></div></div> ✓	

generic data type에 대한 다음 설명 중 맞는 것을 모두 고르시오.

c++ template을 통해서도 구현할 수 있다.	응답자 23 명	96 %	<div><div></div></div> ✓	21 % 정답
일반화된 자료형으로 전체 자료구조를 설계하고 상황마다 필요한 내부 데이터 타입은 재정의하여 사용하는 형태로 구현한다.	응답자 17 명	71 %	<div><div></div></div> ✓	
template을 통해 구현할 경우 데이터 타입은 런타임에 결정된다.	응답자 9 명	38 %	<div><div></div></div>	
사용자 정의형 데이터를 generic data type으로 사용하려면 연산자 오버로딩이 필요할 수 있다.	응답자 14 명	58 %	<div><div></div></div> ✓	

시도 : 24 / 24

생성자에 대한 다음 설명 중 맞는 것을 모두 고르시오.

객체 생성 시 호출될 생성자는 함수 구현된 순서로 결정된다.	응답자 4 명	17 %	<div></div>	21 % 정답
우리가 어떤 생성자를 구현하더라도 기본 생성자는 기본으로 생성된다.	응답자 14 명	58 %	<div></div>	
복사 생성자는 해당 class와 연관하여 =을 사용하는 모든 경우에 호출된다.	응답자 8 명	33 %	<div></div>	
기본 생성자란 아무 파라미터를 가지지 않는 생성자를 의미한다.	응답자 20 명	83 %	<div>✓</div>	

시도 : 24 / 24

-0.21

차별 지수 ⑦

어떤 프로그램이 다음과 같은 복잡도 함수 $f(N)$ 을 가질 때 $f(N)$ 은 $O(N^4)$ 이다.

$$f(N) = N^3 - 200N^2 + 5000$$

참	응답자 4 명	17 %	<div>✓</div>	17 % 정답
거짓	응답자 20 명	83 %	<div></div>	

다음 코드에서 단위 연산을 비교문으로 할 때의 복잡도 함수 $f(N)$ 을 구하시오.

```
void f(int arr[], int N)
{
    for (int i = 0; i < N; i++)
        if (arr[i] > arr[i + 1])
            arr[i] = arr[i + 1];
}
```

O(N)	응답자 4 명	17 %	<div><div></div></div> ✓
$f(N) = O(N)$		0 %	<div><div></div></div> ✓
$f(N) = O(N)$		0 %	<div><div></div></div> ✓
$f(N) = N$	응답자 3 명	13 %	<div><div></div></div> ✓
$f(N) = N$	응답자 2 명	8 %	<div><div></div></div> ✓
N	응답자 3 명	13 %	<div><div></div></div> ✓
다른 것	응답자 11 명	46 %	<div><div></div></div>
답 없음	응답자 1 명	4 %	<div><div></div></div>

50 % 정답

시도 : 24 / 24

스택에 대한 다음 설명 중 맞는 것을 모두 고르시오.

stack의 내용을 모두 pop한 내용을 다른 stack에 push하면 동일한 내용으로 복사가 된다.	응답자 6 명	25 %	<div></div>	42 % 정답
스택은 후입선출(LIFO) 방식으로 동작한다.	응답자 22 명	92 %	<div></div> ✓	
stack의 기본 연산은 push, top, pop, next 등이 있다.	응답자 8 명	33 %	<div></div>	
스택은 linked list를 이용해 구현하는 것이 항상 효율적이다.	응답자 2 명	8 %	<div></div>	



시도 : 24 / 24

&, * 두 연산자에 대한 다음 설명 중 맞는 것을 모두 고르시오.

&변수명 형태로 쓰이면 참조 변수를 선언한 것이다.	응답자 16 명	67 %	<div></div>	8 % 정답
데이터 타입 *변수명 형태로 쓰이면 포인터 변수를 선언한 것이다.	응답자 23 명	96 %	<div></div> ✓	
참조 변수 역시 주소값을 복사하여 넘겨주는 것으로 포인터와 동일하게 동작한다.	응답자 4 명	17 %	<div></div>	
동적 할당을 사용하는 사용자 정의형 자료에서 딥 카피를 할 목적으로 참조 변수를 사용할 수 있다.	응답자 12 명	50 %	<div></div>	

동적할당을 잘못 했을 때 발생하는 대표적인 두 문제 dangling pointer와 메모리 누수에 대한 다음 설명 중 맞는 것을 모두 고르시오.

메모리 누수가 발생하면 런타임 에러가 나면서 바로 프로그램이 종료된다.	응답자 3 명	13 %	<div></div>	17 % 정답
메모리 해제 후 항상 포인터에 NULL을 입력하면 dangling pointer가 발생하는 것을 방지할 수 있다.	응답자 16 명	67 %	<div></div>	
dangling pointer를 방지하기 위해서는 동일한 메모리를 가리키는 포인터를 여러 개 사용할 때 메모리 해체에 주의하여야 한다.	응답자 22 명	92 %	<div>✓</div>	
dangling pointer는 포인터에 NULL이나 다른 메모리 주소를 대입하여 기존 동적할당한 메모리에 접근할 수 없게 되는 상황을 의미한다.	응답자 11 명	46 %	<div></div>	

큐를 서클러 형태로 구현할 때 rear, front 포인터로 일반적인 형태로 하면 어떤 문제가 발생할 수 있고

해당 문제를 어떤 방식으로 해결할 수 있는지 3문장으로 서술하시오.





채점되지 않은 답변

응답자 24 명





100 %

SpeedGrader에서 보기

스택의 내부에 linked list 형태로 데이터를 저장할 때 배열 형태로 구현한 경우와 비교하여 어떤 차이가 있는지 아래 설명 중 맞는 것을 모두 고르시오.

배열 형태로 내부를 구현하면 삽입과 삭제에 효율적이다.	응답자 8 명	33 %	
링크드 리스트 형태로 내부를 구현하면 탐색에 효율적이다.	응답자 6 명	25 %	
링크드 리스트 형태로 구현하는 경우 소멸자가 필요하다.	응답자 17 명	71 %	 ✓
링크드 리스트 형태로 구현하는 경우 사용되지 않고 불필요하게 할당되는 메모리 사용을 줄일 수 있다.	응답자 21 명	88 %	 ✓

큐의 내부에 linked list 형태로 데이터를 저장할 때 배열 형태로 구현한 경우와 비교하여 어떤 차이가 있는지 아래 설명 중 맞는 것을 모두 고르시오.

링크드 리스트 형태일 때 비정렬 큐라면 enqueue 함수에서 조건문은 필요하지 않다.	응답자 10 명	42 %	
링크드 리스트 형태로 서술러 큐를 구현하는 경우 front와 rear 두 개의 내부 포인터가 존재할 필요는 없다.	응답자 12 명	50 %	 ✓
링크드 리스트 형태로 구현된 큐를 서술러 큐로 변환하는 것은 배열로 구현된 걸 변환하는 것보다 변경할 부분이 적다.	응답자 15 명	63 %	 ✓
큐가 배열이나 링크드 리스트 어느 쪽으로 구현된 경우에도 탐색을 하는 복잡도는 동일하다.	응답자 11 명	46 %	

다음은 링크드 리스트를 사용하는 정렬된 자료 구조에서 데이터를 삽입하는 함수이다. 빈칸을 채우시오.

```
template <class ItemType>
void SortedType<ItemType>::InsertItem(ItemType item)
{
    NodeType<ItemType>* new_node;
    NodeType<ItemType>* pred;
    NodeType<ItemType>* location;
    bool moreToSearch;

    location = listData;
    pred = NULL;
    moreToSearch = (location != NULL);

    while (moreToSearch)
    {
        if (location->info < item)
        {
            pred = [a1];
            location = location->next;
            moreToSearch = (location != NULL);
        }
        else
            moreToSearch = false;
```



```
    moreToSearch = false;
```

```
}
```

```
new_node = new NodeType<ItemType>;
```

```
new_node->info = item;
```

```
if ([a2] == NULL)
```

```
{
```

```
    [a3] = listData;
```

```
    listData = new_node;
```

```
}
```

```
else
```

```
{
```

```
    [a4] = location;
```

```
    pred->next = new_node;
```

```
}
```

```
length++;
```

```
}
```

s1 s2 s3 s4

location	응답자 16 명	67 %	<div></div> ✓
답 없음	응답자 6 명	25 %	<div></div>
다른 것	응답자 2 명	8 %	<div></div>



pred	응답자 7 명	29 %	<div><div></div></div> ✓
다른 것	응답자 12 명	50 %	<div><div></div></div>
답 없음	응답자 5 명	21 %	<div><div></div></div>

a1

a2

a3

a4

new_node->next

응답자 8 명

33 %



답 없음

응답자 5 명

21 %



다른 것

응답자 11 명

46 %



a1 a2 a3 a4

new_node->next	응답자 9 명	38 %	<div></div> ✓
다른 것	응답자 10 명	42 %	<div></div>
답 없음	응답자 5 명	21 %	<div></div>

시도 : 16 / 24

더블 링크드 리스트에서 헤더와 트레일러를 사용할 경우 얻을 수 있는 장점을 사용 예를 들어 4문장 이내로 설명하시오.

채점되지 않은 답변

응답자 24 명

100 %

SpeedGrader에서 보기

시도 : 19 / 19

복사 생성자에 대한 다음 설명 중 맞는 것을 모두 고르시오.

옐로우 카피를 방지하려면 구현이 필요하다.

응답자 18 명

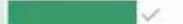
95 %



아무 복사 생성자를 구현하지 않으면 기본 복사 생성자가 생성되며 이 복사 생성자는 옐로우 카피를 수행한다.

응답자 13 명

68 %



복사 생성자만 구현하면 옐로우 카피를 방지하고 항상 딥 카피가 되도록 할 수 있다.

응답자 4 명

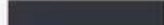
21 %



복사 생성자는 헤더에 파라미터로 자기와 같은 타입의 객체를 사용한다.

응답자 16 명

84 %



답 없음

응답자 5 명

26 %



11 % 정답

시도 : 13 / 24

연산자 오버로딩을 할 때 멤버 함수로 구현하는 경우와 전역 함수로 구현하는 경우 파라미터가 차이가 나는 이유를 3문장 이내로 설명하시오.

채점되지 않은 답변

응답자 24 명

100 %

[SpeedGrader에서 보기](#)