꿈꾸는 라이언

정보처리기사 실기

기출풀이집 - 프로그래밍 언어편

목 차

안녕하세요! **꿈꾸는라이언**입니다.
먼저 정처기 필기 합격을 진심으로 축하드려요! 🎉 🎊

이제 실기를 준비하시는 분들을 위해 **프로그래밍 언어편 요약 노트**를 준비했습니다.

요즘 정처기 실기 시험의 프로그래밍 언어 비중이 매우 높아졌어요. 전체 20문항 중 8~10문제가 프로그래밍 언어 관련 문제라니 절반에 가까운 수준이죠. 😳 따라서 프로그래밍 언어를 제대로 준비하지 않으면 아무리 이론을 잘 알아도 합격이 쉽지 않은게 현실입니다.

실기에서 다루는 프로그래밍 언어는 C언어, 자바, 파이썬인데요, **출제 비중은 C언어(3~4문제) > 자바(2~3문제) > 파이썬(1~2문제) 순**이에요. 그래서 점수의 비중으로는 C언어가 가장 중요하고, 그 다음으로 자바와 파이썬 순이에요!
하지만 이 노트에서는 오히려 정말 유형도 적고 이해하기 쉬운 파이썬부터 시작해서 자바, C언어 순서로 정리했어요.
처음 프로그래밍 언어를 접하시는 분들도 차근차근 따라오실 수 있게요! 😊

1. **프로그래밍 언어에서는 기출 유형이 반복돼요!** 그래서 20년부터 23년도까지의 모든 프로그래밍 언어 기출문제를 분석하여 자주 출제되는 유형을 요약했어요. 단순 암기식 문제부터 응용력을 요구하는 문제까지 다양한 레벨의 문제가 있지만, 결국 같은 유형이 반복되고 있었어요. 이 유형만 확실히 잡으시면 프로그래밍 언어 문제는 충분히 고득점 받으실 수 있을 거예요! 👍

2. **모든 코드를 이해할 필요는 없어요!** 코드 한줄 한줄 이해를 하고 넘어가면 정말 좋지만, 문제 풀이에 있어서는 출제 유형을 빠르게 파악하는 것이 관건입니다. 실제로 많은 문제들이 숫자만 바뀌어서 동일한 유형으로 출제되거든요. 예를 들어 완전수를 찾는 문제가 매번 나오는데, 사실 완전수는 100이하에서는 6과 28뿐이에요. 그런데 문제의 코드를 전부 해석하려 하면 손으로 100번까지 계산해야 할 수도 있어요. 😱 하지만 완전수의 특징만 알고 있다면 30초 안에 답을 찾을 수 있죠. 그러니 모든 코드를 이해하는데 집중하기보다는 먼저 어떤 유형의 문제인지를 캐치하는 것이 효과적인 전략이 될 거예요. 이를 위해 제가 정리한 분석을 통해 반복적으로 출제되는 유형을 정확히 파악하시는 걸 추천드려요!
   추가로 각 기출 문제마다 아래와 같이 난이도를 표시했어요! **낮은 난이도부터 우선해서 학습해주세요!**
   ★☆☆ : 반드시 맞춰야 하는 쉬운 문제
   ★★☆ : 처음에는 다소 어렵지만 합격을 위해서는 도전해야 하는 문제
   ★★★ : 실제 시험에서는 풀지 말아야할 정도로 어려운 문제

   프로그래밍의 실행 순서가 복잡한 문제의 경우, 작은 숫자(①②…)로 코드 진행 순서를 각 코드 앞에 표기해놨어요!

3. **유튜브에 풀이가 있어요!** 모든 문제에 대한 효율적인 풀이 및 코드에 대한 한 줄 한 줄 설명을 적었지만, 이해가 어려운 경우에는 모든 문제에 기출 연도와 회차를 적어두었으니 유튜브에 검색해서 자세한 설명을 보시는 것도 가능해요! 그리고, 프로그래밍 기출 문제는 사람들이 각자 복원한 문제이기에 정리마다 조금씩 코드가 다를 수는 있어요. 하지만, 문제의 유형은 동일하니 걱정하지 않으셔도 됩니다.

이 <정처기 실기 요약노트 - 프로그래밍 언어편>으로 공부하시면서 추가 궁금하신 사항이나 의견 있으실 분들을 위해 **꿈꾸는라이언 공식 카페**에 **'정처기 실기 질의응답 게시판'**을 별도 만들었어요! 카페 가입하시고 질문 남겨주시면 본 요약노트 제작에 많은 도움을 준 현업 개발자 분께서 바로는 아니더라도 틈틈이 답변해주실꺼에요! 😀 👍
문의주신 모든 분들께 최대한 가능한 선에서 성심껏 빠르게 답변해드린다고 하니 참고해주세요!
📌 꿈꾸는라이언 네이버 카페 : https://cafe.naver.com/dreamingryan

그럼 모두들 정처기 합격을 시작으로 각자 계획하셨던 목표와 소중한 꿈을 이루시길 진심으로 응원하겠습니다.
다들 포기하지 마시고, 끝까지 함께 달려봐요!
합격 미리 축하드립니다!! ❤️ 🎉

이것만 이해해도 문제 풀이에 정말 큰 도움이 돼요

★ 집합(Set)

집합은 순서가 없고 중복을 허용하지 않는 자료형이에요.
아래와 같이 사용할 수 있어요.

 # 집합 생성

```
fruit_basket = {"apple", "banana", "grape"}
```

 # 집합에 요소 추가

```
fruit_basket.add("orange")
print(fruit_basket) ⟶ 출력: {"apple", "banana", "grape", "orange"}
```

 # 집합에서 요소 제거

```
fruit_basket.remove("banana")
print(fruit_basket) ⟶ 출력: {"apple", "grape", "orange"}
```

 # 집합 업데이트

```
more_fruits = ["kiwi", "pear", "apple"]
fruit_basket.update(more_fruits)
print(fruit_basket) ⟶ 출력: {"apple", "grape", "orange", "kiwi", "pear"}
```

★ 비트 연산자(Bitwise Operator)

2진수 형태로 데이터를 다룰 때 사용해요.
비트 연산자를 이해하기 쉬운 예시로는 "전등 스위치"를 들 수 있어요.
전등 스위치는 켜짐(1)과 꺼짐(0), 두 가지 상태만 있어요.
*여기서 켜짐은 True, 꺼짐은 False를 의미해요.

AND 연산(&)은 두 개의 스위치가 모두 켜져 있을 때만 전등이 켜지는 거예요.
OR 연산(|)은 두 개의 스위치 중 하나라도 켜져 있으면 전등이 켜지는 거예요.
XOR 연산(^)은 두 개의 스위치 중 하나만 켜져 있을 때 전등이 켜지는 거예요.
(서로의 결과가 달라야 전등이 켜진다는 의미에요)

아래와 같이 사용할 수 있어요. 내가 연산자를 보고 계산할 수 있으면 돼요.

 # AND 연산

```
result = 5 & 3 ⟶ 101 & 011 = 001
print(result) ⟶ 출력: 1
```

 # OR 연산

```
result = 5 | 3 ⟶ 101 | 011 = 111
print(result) ⟶ 출력: 7
```

 # XOR 연산

```
result = 5 ^ 3 ⟶ 101 ^ 011 = 110
print(result) ⟶ 출력: 6
```

 # 비트 시프트 연산

왼쪽으로 한칸 이동은 맨 오른쪽에 0하나를 추가한다는 의미에요.

```
result = 5 << 1 ⟶ 101 << 1 = 1010
print(result) ⟶ 출력: 10
```

오른쪽으로 한칸 이동은 맨 오른쪽 값을 지운다는 의미에요.
여기서 2진수 010과 10은 같은 2를 뜻해요.

```
result = 5 >> 1 ⟶ 101 >> 1 = 010
print(result) ⟶ 출력: 2
```

★ split()

문자열을 지정된 구분자를 기준으로 분할하여 리스트로 반환합니다.
기본 구분자는 공백입니다.

 # 공백을 기준으로 문자열 분할

```
sentence = "Hello world! How are you?"
words = sentence.split() ⟶ sentence.split(" ") 과 동일한 표현
print(words) ⟶ 출력: ['Hello', 'world!', 'How', 'are', 'you?']
```

★ 슬라이싱(Slicing)

리스트나 문자열에서 일부분을 잘라내는 것을 말해요.
Index의 시작점은 0부터 시작해요.
끝점의 index는 자를 때 포함하지 않아요.
[0:5] 는 첫 번째 문자부터 시작해서 0, 1, 2 ,3, 4까지의 문자를 뜻해요.

처음부터 자르고 싶으면 시작점을 생략할 수 있어요.
[:5]
끝까지 자르고 싶으면 끝점을 생략할 수 있어요.
[8:]

 # 문자열 슬라이싱

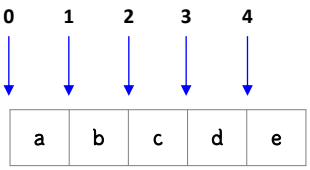
```
message = "Hello, World!"
print(message[0:5]) ⟶ 출력: "Hello"
print(message[:5]) ⟶ 출력: "Hello"
print(message[7:]) ⟶ 출력: "World!"
print(message[7:12]) ⟶ 출력: "World"
print(message[::2]) ⟶ 출력: "Hlo ol!" → 홀수번째 문자열 추출!
```

(2 page 2번 문제 참고)

 # 리스트 슬라이싱

```
numbers = ["a", "b", "c", "d", "e"]
print(numbers[1:4]) ⟶ 출력: ["b", "c", "d"]
print(numbers[:3]) ⟶ 출력: ["a", "b", "c"]
print(numbers[3:]) ⟶ 출력: ["d", "e"]
print(numbers[::2]) ⟶ 출력: ["a", "c", "e"]
```

→ 홀수번째 문자열 추출!
(2 page 2번 문제 참고)



numbers[1:4] 슬라이싱 시,
인덱스 1에서 시작해서
인덱스 4 앞까지 추출
(**인덱스 3**까지만 포함)

★ 문자열 포맷팅 (String Formatting)

문자열 내에 특정 값을 삽입하는 방법이에요.
% 연산자를 사용하여 문자열 포맷팅을 할 수 있어요.

 # % 연산자

```
name = "Alice"
age = 25

print("My name is %s. I'm %d years old." % (name, age))
My name is Alice. I'm 25 years old.
```

★ map()

리스트나 튜플 등의 이터러블 객체의 각 요소에 함수를 적용하여 새로운 이터러블 객체를 반환합니다.
map()은 원본 데이터를 변경하지 않고 새로운 데이터를 생성합니다.

numbers = [1, 2, 3, 4, 5]

 # lambda 함수를 사용하여 각 요소에 10을 더하기

```
numbers = [1, 2, 3, 4, 5]
add_ten = map(lambda x: x + 10, numbers)
print(list(add_ten)) ⟶ 출력: [11, 12, 13, 14, 15]
```

[기출 예제] 23년 3회 실기 (난이도: ★☆☆)

다음 Python 프로그램과 그 <실행결과>를 분석하여 괄호에 들어갈 알맞은 예약어를 쓰시오.
첫 번째 라인의 '5 10'은 입력받은 값에 해당한다.
<실행결과>
x, y의 값을 공백으로 구분하여 입력: 5 10
x의 값: 5
y의 값: 10

▶ 정답 split

정답은 split 이에요
split() 메서드는 문자열을 특정 구분자를 기준으로 나누어 리스트로 반환해요.
기본적으로는 공백을 구분자로 사용해요.

split() 메서드는 사용자 입력을 받아 분리할 때 자주 사용돼요.

```
1  x, y = input("x, y의 값을 공백으로 구분하여 입력: ").( 빈 칸 )(" ")
2  print("x의 값: ", x)
3  print("y의 값: ", y)
```

input() 함수는 사용자로부터 입력을 받는 기능을 해요.
"x, y의 값을 공백으로 구분하여 입력: "은
사용자에게 무엇을 입력해야 하는지 알려주는 안내 문구예요.

★ key point
사용자가 5 10 입력한 것을 실행결과에서 알려주고 있어요.
"5 10"이라는 값을 x, y로 공백 기준으로 값을 할당하기 위해서는 공백 기준으로 문자열을 나눠야해요.
문자열을 특정 구분자로 나누는 키워드는 split이에요.
그래서 정답이 split이 돼요.

*구분자를 넘겨주지 않는 split() 은 split(" ") 과 동일해요. 즉, 공백 문자열이 기본값이에요.

[기출 예제] 23년 2회 실기 (난이도: ★☆☆)

다음 파이썬 코드에서 알맞는 출력값을 작성하시오.

▶ 정답 engneing

```
1  a = "engineer information processing"
2  b = a[:3]
3  c = a[4:6]
4  d = a[28:]
5  e=b+c+d
6  print(e)
```

변수 a에 문자열 "engineer information processing"를 할당해요.

a 문자열의 처음부터 3번째 문자까지 가져와 b에 저장합니다.
여기서 a[:3]은 a의 인덱스 0부터 인덱스 2까지를 의미해요.
따라서 b에는 "eng"가 저장됩니다.

a 문자열에서 5번째부터 6번째 문자까지 가져와 c에 저장합니다.
(인덱스 4부터 인덱스 5까지)
여기서 a[4:6]은 "ne"을 의미해요.

변수 d에 a의 29번째 문자부터 끝까지의 슬라이스를 할당해요.
슬라이싱에서 두 번째 인덱스를 생략하면 문자열의 끝까지 슬라이스 해요.
a의 29번째 문자부터 끝까지는 "ing" 부분 문자열 이에요.
따라서 d에는 "ing"이 할당돼요.

변수 e에 b, c, d를 연결한 문자열을 할당해요.
b는 "eng", c는 "ne", d는 "ing"이므로,
e에는 " engneing"이 할당돼요.

[6행] 변수 e의 값인 " engneing"을 출력해요.

★ 슬라이싱 (자주 쓰이니 외우면 좋아요)
sequence[start:end:step]의 형태로 사용돼요.

start는 슬라이스가 시작되는 인덱스이고,
end는 슬라이스가 끝나는 인덱스이며,
(해당 인덱스는 문자열에서 제외)
step은 슬라이스가 진행되는 간격이에요.

start와 end는 생략할 수 있고, 생략하면 각각
시퀀스의 시작과 끝이 기본값으로 사용돼요.
step도 생략할 수 있고,
생략하면 기본값 1이 사용돼요.

★ 집합 (자주 쓰이니 외우면 좋아요)
중복되지 않는 요소들의 모음이에요.
같은 값을 여러 번 추가해도 집합에는 한 번만 저장돼요.

순서가 없어요.
집합의 요소들은 특정한 순서를 가지지 않아요.

[기출 예제] 23년 1회 실기 (난이도: ★☆☆)

다음 파이썬 코드에서 알맞는 출력값을 작성하시오.

▶ 정답 {'한국', '중국', '베트남', '홍콩', '태국'}

```
1  a = {'한국', '중국', '일본'}
2  a.add('베트남')
3  a.add('중국')
4  a.remove('일본')
5  a.update({'홍콩', '한국', '태국'})
6  print(a)
```

a라는 이름의 집합을 생성하고 초기값으로 '한국', '중국', '일본'을 갖게 해요.

a 집합에 '베트남'을 추가해요.
집합에 새로운 요소를 추가할 때는 add() 메서드를 사용해요.
이 연산 후에 a는 {'한국', '중국', '일본', '베트남'}이 돼요.

a 집합에 '중국'을 추가하려고 해요.
하지만 '중국'은 이미 a에 존재하는 요소예요.
집합은 중복된 요소를 허용하지 않기 때문에
이 연산은 a를 변화시키지 않아요.
이 연산 후에도 a는 여전히 {'한국', '중국', '일본', '베트남'}이에요.

a 집합에서 '일본'을 제거해요.
집합에서 특정 요소를 제거할 때는 remove() 메서드를 사용해요.
이 연산 후에 a는 {'한국', '중국', '베트남'}이 돼요.

최종적인 a 집합을 출력해요.
정답은 {'한국', '중국', '베트남', '홍콩', '태국'}이에요

a 집합에 다른 집합의 요소들을 추가해요.
update() 메서드는 인자로 주어진 집합의 모든 요소를 a에 추가해요.
'한국'은 이미 a에 존재하므로 중복 추가되지 않아요.

[기출 예제] 22년 3회 실기 (난이도: ★★☆)

▶ 정답 [101,102,103,104,105]

다음 파이썬 코드에 대한 출력값을 작성하시오.

```
1 TestList = [1,2,3,4,5]
2 TestList = list(map(lambda num : num + 100, TestList))
3
4 print(TestList)
```

TestList라는 이름의 리스트를 생성하고 초기값으로 [1,2,3,4,5]를 할당해요.

결과적으로, TestList의 각 요소에 100을 더한 새로운 리스트를 생성해요.

map() 함수는 첫 번째 인자로 함수를, 두 번째 인자로 이터러블(리스트 등)을 받아요.
map() 함수는 **이터러블의 각 요소에 함수를 적용한 결과를 반환**해요.

여기 코드에서는 람다 표현식 lambda num : num + 100이 map() 함수의 첫 번째 인자로 사용되었어요.
이 람다 표현식은 인자 num을 받아 num + 100을 반환하는 함수예요.
map()의 두 번째 인자로는 TestList가 전달되었어요.

따라서, map()은 TestList의 각 요소에 람다 함수를 적용해요.
1 + 100 = 101
2 + 100 = 102
3 + 100 = 103
4 + 100 = 104
5 + 100 = 105

map()의 결과는 맵 객체(map object)로 리스트로 변환하기 위해 list() 함수를 사용했어요.
최종적으로 TestList에는 [101, 102, 103, 104, 105]가 할당돼요.

정답은 [101, 102, 103, 104, 105] 이 출력돼요.

[기출 예제] 22년 2회 실기 (난이도: ★★☆)

▶ 정답 REMEMBER AND STR

다음 파이썬 코드에 대한 출력값을 작성하시오.

```
1 a = "REMEMBER NOVEMBER"
2 b = a[:3] + a[12:16];
3 c = "R AND %s" % "STR";
4 print(b+c);
```

a라는 변수에 문자열 "REMEMBER NOVEMBER"를 할당해요.

문자열 슬라이싱을 사용하여 새로운 문자열을 만들어 b에 할당하는 부분이에요.

a[:3]은 a의 처음부터 3번째 문자까지의 슬라이스로, "REM"이에요.
(인덱스 0부터 인덱스 2까지)
a[12:16]은 a의 13번째 문자부터 16번째 문자까지의 슬라이스로, "EMBE"예요.
(인덱스 12부터 인덱스 15까지)

두 슬라이스를 **+ 연산자로 연결하면 "REMEMBE"**가 돼요.
따라서 b에는 "REMEMBE"가 할당돼요.

문자열 포맷팅을 사용하여 새로운 문자열을 만들어 c에 할당하는 부분이에요.
"R AND %s"는 포맷 문자열로, %s 부분에 다른 문자열이 들어갈 자리예요.
% 연산자 뒤에 오는 "STR"이 %s 자리에 들어가요.
따라서 c에는 "R AND STR"이 할당돼요.

b와 c를 연결한 문자열을 출력하는 부분이에요.
b는 "REMEMBE", c는 "R AND STR"이에요.
두 문자열을 + 연산자로 연결하면 "REMEMBER AND STR"이 돼요.

정답은 REMEMBER AND STR 이에요.

[기출 예제] 22년 1회 실기 (난이도: ★★☆)

▶ 정답 a= 20 b= 2

다음 파이썬 코드에 대한 출력값을 작성하시오.

```
1 def exam(num1, num2=2):
2   print('a=', num1, 'b=', num2)
3 exam(20)
```

이 코드의 특징은 함수의 매개변수에 기본값을 지정하는 방법을 보여줘요.

함수 정의에서 **num2=2와 같이 매개변수에 =로 값을 할당**하면, **그 값이 해당 매개변수의 기본값**이 돼요.
함수를 호출할 때 해당 매개변수에 값을 전달하지 않으면, 기본값이 사용돼요.

여기서는, exam(20)은 num1에 20을 전달하고 num2에는 아무것도 전달하지 않았어요
따라서, num1은 20, num2는 기본값 2가 사용돼요.

그러므로 이 코드의 출력 결과는 a= 20 b= 2 에요.

[기출 예제] 21년 3회 실기 (난이도: ★☆☆)

다음 파이썬 코드에 대한 출력값을 작성하시오.

▶ 정답 False

```
1  a,b = 100, 200
2  print(a==b)
```

a에 100을, b에 200을 할당하는 거예요.
이렇게 하면 a와 b에 각각 다른 값이 저장돼요.

a와 b가 같은지 확인하는 거예요.

==은 두 값이 같은지 비교하는 연산자예요.
만약 a와 b가 같다면 True를,
다르다면 False를 결과로 돌려줘요.

a는 100, b는 200이니까
당연히 둘은 같지 않아요.
그래서 a==b의 결과는 False가 되는 거죠.

★ 연산자 종류

관계	==	같다	>	크다	<	작다
	!=	같지 않다	>=	크거나 같다	<=	작거나 같다
비트	&	and	모든 비트가 1이면 1			
	^	xor	모든 비트가 같으면 0, 하나라도 다르면 1			
	|	or	모든 비트 중 하나라도 1이면 1			
	~	not	각 비트의 부정 (0이면 1 산출)			
	<<	-	비트를 왼쪽으로 이동			
	>>	-	비트를 오른쪽으로 이동			
논리	!	not	&&	and	||	or

[기출 예제] 21년 2회 실기 (난이도: ★★★)

다음 파이썬 코드에 대한 출력값을 작성하시오.

▶ 정답 26

```
1  a = 100
2  result = 0
3  for i in range(1,3):
4     result = a >> i
5     result = result + 1
6  print(result)
```

a에 100을 할당하고,
result에 0을 할당했어요.
이렇게 하면 a는 100, result는 0으로 초기화되는 거예요.

for 루프를 사용해서 i를 1부터 2까지 반복해요.
range(시작 숫자, 끝 숫자) 형태를 사용하는데, 이때 끝 숫자는 포함되지 않습니다.
슬라이스와 비슷하게 인덱스 시작 숫자 이상 인덱스 끝 숫자 – 1 이하 까지를 뜻해요.

루프 안에서는 아래 로직을 수행해요.

result = a >> i는 a를 오른쪽으로 i번 시프트한 결과를 result에 할당해요.
>>는 비트 오른쪽 시프트 연산자예요.
이 연산자는 a의 비트를 오른쪽으로 i번 이동시켜요.

조금 어려울 수 있어요.
첫 번째 반복(i=1)에서 a >> 1은 a의 비트를 오른쪽으로 1번 이동시키는 거예요.
2진수로 표현하면 1100100(기존 100)에서 0110010(= 110010)(비트 이동)이 되죠.
이것은 10진수로 50이에요.
여기에 1을 더하면 51이 돼요.

두 번째 반복(i=2)에서는 a >> 2로 a의 비트를 오른쪽으로 2번 이동시켜요.
2진수로 1100100(100)에서 0011001(= 11001)이 되는데, 이것은 10진수로 25예요.
여기에 1을 더하면 26이 돼요.

루프가 끝나면 print(result)로 최종 result의 값을 출력해요.
정답은 26이에요.

★ key point
비트 연산자가 나오면, 2진수로 변환 후
왼쪽 혹은 오른쪽으로 이동 후 다시
10진수로 변환한다고 생각하면 돼요.

[기출 예제] 21년 1회 실기 (난이도: ★☆☆)

다음 파이썬 코드에 대한 출력값을 작성하시오.

▶ 정답 skiddp

```
1  class good :
2      li = ["seoul", "kyeonggi","inchon","daejeon","daegu","pusan"]
3
4  g = good()
5  str01 = ''
6  for i in g.li:
7      str01 = str01 + i[0]
8
9  print(str01)
```

good 클래스의 인스턴스를 생성해서 g라는 변수에 할당하는 거예요.
이렇게 하면 g를 통해 good 클래스의 속성과 메서드에 접근할 수 있게 돼요.

빈 문자열을 str01이라는 변수에 할당하는 거예요.
이 변수는 앞으로 도시 이름의 첫 글자를 모으는 데 사용될 거예요.

for 루프를 사용해서 g.li의 각 요소를 i라는 변수에 하나씩 할당하면서 반복해요.
여기서는 순서대로, seoul, kyeonggi … 이 값들이 i에 할당돼요.

str01 = str01 + i[0]은 현재 str01의 값에 i의 첫 번째 문자(i[0])를 더하는 거예요.
문자열에서 배열처럼 index로 문자 하나씩 가져올 수 있어요.

첫 번째 반복에서는 "seoul"의 첫 글자 "s"가 str01에 더해져요.
두 번째 반복에서는 "kyeonggi"의 첫 글자 "k"가 str01에 더해져요.
이런 식으로 계속 반복되면서 str01에는 각 도시 이름의 첫 글자가 모이게 돼요.

루프가 끝나면 print(str01)로 최종 str01의 값을 출력해요.
정답은 skiddp 이에요.

▶ 정답
[1, 2, 3]
7
123
45
6789

[기출 예제] 20년 4회 실기 (난이도: ★★★)

다음 파이썬 코드에 대한 출력값을 작성하시오.

```
1  lol = [[1,2,3],[4,5],[6,7,8,9]]
2  print(lol[0])
3  print(lol[2][1])
4  for sub in lol:
5      for item in sub:
6          print(item, end = '')
7      print()
```

리스트들을 요소로 가지고 있는 2차원 리스트예요.

lol의 첫 번째 요소를 출력하는 거예요. lol[0]은 [1,2,3]이므로, 이 코드는 [1,2,3]을 출력해요.

세 번째 요소의 두 번째 요소를 출력하는 거예요.
lol[2]는 [6,7,8,9]이고, lol[2][1]은 7이므로, 이 코드는 7을 출력해요.

중첩 for 루프를 사용하고 있어요.
바깥쪽 루프에서는 lol의 각 요소(리스트)를 sub라는 변수에 하나씩 할당하면서 반복해요.

안쪽 루프에서는 각 sub의 요소를 item이라는 변수에 하나씩 할당하면서 반복해요.

현재 item의 값을 출력하는데,
end = ''는 출력 후 줄바꿈을 하지 않고 바로 다음 출력을 이어서 하라는 의미예요.
이렇게 하면 각 리스트의 요소들이 한 줄에 모두 출력돼요.

안쪽 루프가 끝나면 print()를 사용해서 줄바꿈을 해요.
이렇게 하면 각 리스트의 요소들이 한 줄씩 출력되는 거죠.
item 기준으로는 한줄로 출력하고,
Sub 기준으로는 한줄씩 내려서 출력돼요.

item에 이라는 파이썬 키워드가 있는 건 아니에요.
lol, sub과 동일한 변수명이에요!

첫번째 배열인 [1,2,3] 는 123
두번째 배열인 [4,5] 는 45
세번째 배열인 [6,7,8,9] 는 6789
이렇게 출력돼요.

[기출 예제] 20년 2회 실기 (난이도: ★★☆)

다음 파이썬 코드에 대한 출력값을 작성하시오.

▶ 정답 {'중국', '한국', '베트남', '홍콩', '태국'}

```
1  a={'일본','중국','한국'}
2  a.add('베트남')
3  a.add('중국')
4  a.remove('일본')
5  a.update(['홍콩','한국','태국'])
6  print(a)
```

집합(set) 자료형을 다루고 있어요. 집합은 순서가 없고, 중복된 요소를 허용하지 않는 자료형이에요.
자주 나오고 있어요. 중복되지 않는다는 사실 알고 있으면 좋아요.
중괄호 {}를 사용하여 생성할 수 있고, 빈 집합을 생성할 때는 set()을 사용해요.

a 집합에 '베트남'을 추가하는 거예요. 이 연산 후에 a는 {'일본', '중국', '한국', '베트남'}이 돼요.

a 집합에 '중국'을 추가하려고 하는 거예요. 하지만 '중국'은 이미 a에 존재하는 요소예요.
집합은 중복된 요소를 허용하지 않기 때문에, 이 연산은 a를 변화시키지 않아요.

a 집합에서 '일본'을 제거하는 거예요. 이 연산 후에 a는 {'중국', '한국', '베트남'}이 돼요.

a 집합에 '홍콩', '한국', '태국'을 추가하는 거예요.
update() 메서드는 다른 집합이나 리스트 등의 이터러블 객체를 인자로 받아서,
그 안의 모든 요소를 집합에 추가해요. 이 때, '한국'은 이미 a에 존재하므로 중복 추가되지 않아요.
이 연산 후에 a는 {'중국', '한국', '베트남', '홍콩', '태국'}이 돼요.

최종적인 a 집합을 출력하는 거예요.
{'중국', '한국', '베트남', '홍콩', '태국'}이 출력돼요.

▶ 정답
Seynaau

[기출 예제] 24년 1회 실기 (난이도: ★★☆)

다음 파이썬 코드에 대한 출력값을 작성하시오.

```
1  a = ["Seoul", "Kyeonggi", "Incheon", "Daejun", "Daegu", "Pusan"]
2  str = "S"
3
4  for i in a:
5      str = str + i[1]
6
7  print(str)
```

문자열을 요소로 가지고 있는 리스트예요.
str 변수를 생성 후 S를 할당했어요.

1번 라인에서 선언한 배열의 문자열들을 한번씩 i에 할당해요.
변수 str에 str 과 문자열 i의 index 1 위치의 문자를 더해요.
(index는 0부터 시작해요. index 1은 2번째 문자를 뜻해요.)

n[0]	n[1]	n[2]	n[3]	n[4]
S	e	o	u	l

a = ["Seoul", "Kyeonggi", "Incheon", "Daejun", "Daegu", "Pusan"]
즉, S + e + y + n + a + a + u = Seynaau 이 출력돼요.

[기출 예제] 24년 2회 실기 (난이도: ★★☆)

▶ 정답 ab3ca3

다음은 Python에 대한 문제이다. 아래 코드를 읽고 알맞는 출력 값을 작성하시오.

```
1  def fnCalculation(x,y):
2      result = 0;
3      for i in range(len(x)):
4          temp = x[i:i+len(y)]
5          if temp == y:
6              result += 1;
7      return result
8
9  a = "abdcabcabca"
10 p1 = "ab";
11 p2 = "ca";
12
13 out = f"ab{fnCalculation(a,p1)}ca{fnCalculation(a,p2)}"
14 print(out)
```

- result라는 변수를 만들고 0으로 초기화해요. 이 변수는 나중에 결과를 저장할 거예요.
- x의 길이만큼 반복하는 루프를 시작해요. i는 0부터 x의 길이-1까지 변해요.
- x의 일부분을 잘라서 temp에 저장해요. i부터 시작해서 y의 길이만큼의 부분 문자열을 만들어요. ab, bd, dc…ca까지 y와 같은지 체크해요.
- 만약 temp가 y와 같다면, result를 1 증가시켜요. 이렇게 해서 x 안에 y가 몇 번 나타나는지 세고 있어요.
- 최종적으로 계산된 result 값을 반환해요.
- 세 개의 문자열 변수 a, p1, p2를 정의하고 값을 할당해요.
- fnCalculation 함수를 두 번 호출하고, 그 결과를 문자열 안에 넣어 새로운 문자열을 만들어요.
  f-string을 사용해서 결과를 문자열에 직접 삽입하고 있어요.
  ※ 파이썬의 포맷팅 문자열로, 문자열 안에 직접 변수나 표현식을 넣을 수 있어요.

즉, abdcabcabca에 ab가 몇 번 나타나는지 체크한 뒤 3을 반환해요.
abdcabcabca에 ca가 몇 번 나타나는지 체크한 뒤 3을 반환해요.
그래서 ab3ca3이 출력돼요.

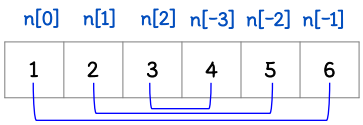
[기출 예제] 24년 3회 실기 (난이도: ★★☆)

▶ 정답 3

다음은 파이썬에 대한 문제이다. 아래 코드를 확인하여 알맞는 출력값을 작성하시오.

```
1  def func(lst):
2      for i in range(len(lst) // 2):
3          lst[i], lst[-i - 1] = lst[-i - 1], lst[i]
4
5  lst = [1, 2, 3, 4, 5, 6]
6  func(lst)
7  print(sum(lst[::2]) - sum(lst[1::2]))
```

- 리스트 길이의 절반인 3(len(lst) // 2)만큼 반복해요. (// 은 몫만 계산하는 연산자)
- 리스트의 앞쪽 값(lst[i])과 뒤쪽 값(lst[-i-1])을 서로 바꿔요.
  i = 0이면 첫 번째 값(lst[0])과 마지막 값(lst[-1])을 바꿉니다.
- 리스트 lst를 입력받아, 리스트를 뒤집어요. (결과: [6, 5, 4, 3, 2, 1])
- 리스트에서 짝수 인덱스(lst[::2])와 홀수 인덱스(lst[1::2]) 값을 각각 더하고, 그 차이를 계산해요.



i[0] 과 i[-1]의 값을 바꿔요.
i[1] 과 i[-2]의 값을 바꿔요.
i[2] 와 i[-3]의 값을 바꿔요.

lst[::2]: 리스트의 짝수 인덱스 값 [6, 4, 2] (0, 2, 4번째 값) 0부터 2씩 증가하는 인덱스
lst[1::2]: 리스트의 홀수 인덱스 값 [5, 3, 1] (1, 3, 5번째 값) 1부터 2씩 증가하는 인덱스
sum(lst[::2]): 6 + 4 + 2 = 12
sum(lst[1::2]): 5 + 3 + 1 = 9

차이: 12 - 9 = 3 이 정답이에요.

★ key point
배열의 뒤집기와 **리스트 슬라이싱**에 대해서만 알고 있다면 어렵지 않게 풀 수 있어요.

★ key point
type 함수의 정확한 이해와 조건문의 논리적인 흐름을 아는지 확인하는 문제에요.

[기출 예제] 24년 3회 실기 (난이도: ★★☆)

▶ 정답 45

다음은 파이썬에 대한 문제이다. 아래 코드를 확인하여 알맞는 출력값을 작성하시오.

```
1  def func(value):
2      if type(value) == type(100):
3          return 100
4      elif type(value) == type(""):
5          return len(value)
6      else:
7          return 20
8
9
10 a = '100.0'
11 b = 100.0
12 c = (100, 200)
13
14 print(func(a) + func(b) + func(c))
```

- 먼저 value의 type과 100의 type 즉, 정수(int) 타입인지 확인해요. 맞다면, 100을 반환해요.
- 정수 타입이 아니라면 문자열(str) 타입인지 확인해요. 맞다면 그 문자열 길이를 반환해요.
- 문자열 타입이 아니라면 20을 반환해요.

a는 문자열 "100.0" 이에요. 따라서, type(value) == type("")의 조건이 참이에요.
문자열 길이 5를 반환해요.

b는 실수형 숫자 100.0 이에요. 즉, 정수 타입과 문자열 타입 모두 아니기 때문에 20을 반환해요.

c는 튜플 (100, 200) 이에요. b와 동일하게 정수 타입과 문자열 타입 모두 아니기 때문에 20을 반환해요.

- func(a) → 5
- func(b) → 20
- func(c) → 20

으로 5 + 20 + 20 = 45가 출력돼요.

▶ 정답 13

[기출 예제] 25년 1회 실기 (난이도: ★★☆)

다음은 파이썬에 대한 문제이다. 아래 코드를 확인하여 알맞는 출력값을 작성하시오.

```
1  class Node:
2      def __init__(self, value):
3          self.value = value
4          self.children = []
5
6  def tree(li):
7      nodes = [Node(i) for i in li]
8      for i in range(1, len(li)):
9          nodes[(i - 1) // 2].children.append(nodes[i])
10     return nodes[0]
11
12 def calc(node, level=0):
13     if node is None:
14         return 0
15     return (node.value if level % 2 == 1 else 0) + sum(calc(n, level + 1) for n in node.children)
16
17 li = [3, 5, 8, 12, 15, 18, 21]
18
19 root = tree(li)
20
21 print(calc(root))
```

노드를 만들 때 실행되는 초기화 함수에요.
value라는 값을 받아서, **노드 안에 저장**하고 **자식 노드들을 담을 빈 리스트**도 함께 가지고 있어요.

리스트 li의 각 값을 이용해 Node 객체들을 순서대로 만들어요. ex) li = [3,5,8,12,15,18,21]라면, [Node(3), Node(5), ..., Node(21)]

인덱스 i의 노드를, 인덱스 (i-1)//2의 노드의 자식으로 연결해요.

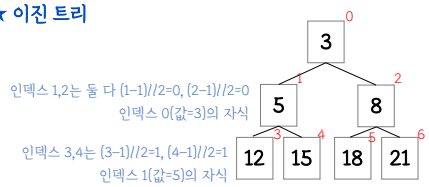
이 규칙 덕분에 리스트가 완전 이진 트리로 변경돼요.
(왼쪽부터 빈 자리 없이 채워지는 형태)
※ //는 나눗셈한 몫을 반환해요 ex) 7 // 2 = 3

재귀 종료

현재 레벨이 홀수면 node.value를, 짝수면 0을 더해요.
이 노드 자식들에 대해, (level+1) 하여 calc를 호출한 결과를 전부 더해요.

li로 트리를 만들어요.

★ 이진 트리



calc(root)를 호출하여, 홀수 레벨의 노드 값 합을 계산한 뒤 출력해요.
레벨 0 (짝수): 3 → 더하지 않음
레벨 1 (홀수): 5, 8 → 더함 → 5 + 8 = 13
레벨 2 (짝수): 12, 15, 18, 21 → 더하지 않음
더 이상 자식이 없으므로 재귀가 끝나고, 최종 합계는 13입니다.

★ key point

파이썬 문제임을 감안할 때, 별 2.5개 정도로 꽤 어려운 문제입니다.

- tree() 함수로 **완전 이진 트리**를 만든다는 점
- calc() 함수에서 **홀수 레벨의 값만 더한다**는 로직
- 그리고 **클래스 구조** · **리스트 인덱스 연산** · **재귀 호출**의 동작 원리

이 세 가지를 이해하고 나면, 작은 리스트에서는 일일이 값을 대입해 확인하지 않아도 논리만으로 쉽게 풀 수 있습니다.
물론 아직 익숙하지 않다면, 실제 시험에서는 전체 리스트를 전부 써 보면서 동작 방식을 확인해 보는 것도 좋은 방법입니다.
(리스트가 작을 때는 오래 걸리지 않기 때문입니다. 만약 대입할 값이 많다면 다른 문제부터 푸는 것을 추천해 드립니다.)

[문제 유형] equals

★ == 연산자

==는 기본 데이터 타입(예: int, float, char 등)의 값을 비교할 때 자주 사용돼요. 두 값이 같은지 확인하려면 ==를 사용하면 되죠.

참조 데이터 타입(예: 객체)에 대해서 ==를 사용하면, 그것들이 같은 메모리 주소를 가리키고 있는지 확인합니다. 즉, 두 변수가 같은 객체를 참조하고 있는지를 봐요.

```
String text1 = new String("hello");
String text2 = new String("hello");
System.out.println(text1 == text2); // false, 왜냐하면 메모리 주소가 다르니까요.
```

여기서 text1과 text2는 같은 내용의 문자열을 가지고 있지만, 각기 다른 객체이므로 == 연산자는 false를 반환합니다.

★ equals 메서드

equals 메서드는 객체의 내용이 같은지를 확인하는 데 사용되요.

예를 들어, String 클래스에서 equals 메서드는 문자열의 내용이 같은지를 비교합니다.

```
String text1 = new String("hello");
String text2 = new String("hello");
System.out.println(text1.equals(text2)); // true, 내용이 같기 때문이에요.
```

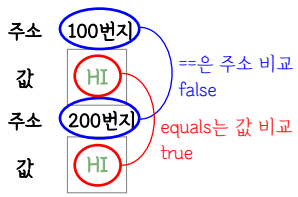
여기서 text1과 text2의 내용은 같기 때문에 equals 메서드는 true를 반환해요.

간단히 말하면, ==는 두 객체가 실제로 같은 객체인지, 즉 메모리 상에서 같은 위치를 참조하는지를 확인하는 반면, equals 메서드는 두 객체가 '논리적으로' 동등한지를 비교하는 데 사용돼요.

[기출 예제] 23년 2회 실기 (난이도: ★☆☆)

다음은 자바에 대한 문제이다. 알맞은 출력 값을 작성 하시오.

```
1  public class Main{
2      public static void main(String[] args) {
3
4          String str1 = 'Programming';
5          String str2 = 'Programming';
6          String str3 = new String('Programming');
7
8
9          println(str1==str2)
10         println(str1==str3)
11         println(str1.equals(str3))
12         print(str2.equals(str3))
13     }
14 }
```



▶ 정답
true
false
true
true

[4행] [5행] 이 둘은 하나의 객체가 생성되고 자바에서 같은 객체를 할당해줍니다.

이렇게 명시적으로 new 를 통해 객체를 생성하는 경우,
같은 값("Programming") 이더라도 별도의 공간에 객체를 생성해 str3에 할당합니다.

str1와 str2의 객체 주소는 동일해요

str1와 str3의 객체 주소는 달라요. new를 통해 새롭게 생성했기 때문이에요.

str1과 str3의 값을 비교해요. 둘의 문자열은 동일해요

str2과 str3의 값을 비교해요. 둘의 문자열은 동일해요

[문제 유형] 단순 for문

★ for 문

for 문은 일정한 규칙에 따라 반복해서 명령을 실행할 때 사용해요.
for 문은 보통 세 부분으로 구성되어 있어요

초기화: 반복을 시작하기 전에 처음에 설정해야 하는 부분이에요. 여기서 반복할 때 사용할 변수를 설정합니다.
조건: 반복을 계속할지 말지를 결정하는 조건이에요. 이 조건이 참이면 반복을 계속하고, 거짓이면 반복을 멈춰요.
증감: 반복할 때마다 실행되는 부분으로, 보통 반복하는 변수를 증가시키거나 감소시키는데 사용돼요.

```
for(int i = 0; i < 5; i++) {
    System.out.println("반복 중: " + i);
}

// 반복 중: 0
// 반복 중: 1
// 반복 중: 2
// 반복 중: 3
// 반복 중: 4
```

`int i = 0;` - 반복을 시작하기 전에 변수 i를 0으로 초기화해요.
`i < 5;` - i가 5보다 작은 동안에는 반복을 계속해요. i가 5에 도달하면 반복을 멈추죠.
`i++` - 매 반복마다 i의 값을 1씩 증가시켜요.

결과적으로 이 for 문은 "반복 중: 0"부터 "반복 중: 4"까지 총 5번의 메시지를 출력하게 됩니다.

[기출 예제] 20년 1회 실기 (난이도: ★☆☆)

▶ 정답 0 1 2 3

다음은 자바 소스코드이다. 출력 결과를 쓰시오.

실제 실행코드

```
1  public class Main {
2      public static void main(String[] args) {
3          int i;
4          int a[] = {0,1,2,3};
5          for(i=0; i<4; i++) {
6              System.out.print(a[i] + " ");
7          }
8      }
9  }
```

4 크기인 배열 a를 생성했어요.
i가 0,1,2,3 까지 반복해요. (4보다 작을 때까지)
a 배열의 0번째, 1번째, 2번째, 3번째 값을 출력해요.
차례대로 **0 1 2 3**이 출력돼요

[기출 예제] 22년 3회 실기 (난이도: ★☆☆)

▶ 정답 0123

아래 자바 코드에서 출력되는 값을 작성하시오.

실제 실행코드

```
1  class Test {
2    ③static int[] mkarr() {
3          int[] tmpArr = new int[4];
4
5          for (int i = 0; i < tmpArr.length; i++) {
6              tmpArr[i] = i;
7          }
8
9          return tmpArr;
10     }
11
12     public static void main(String[] args) {
13   ②     int[] arr;
14         arr = mkarr();
15         for (int i = 0; i < arr.length; i++) {
16             System.out.print(arr[i]);
17         }
18     }
19 }
```

크기가 4인 정수형 배열 tmpArr를 선언하고 생성해요.

0부터 크기가 4인 정수형 배열 의 길이보다 작을 때(4미만)까지 반복
0번 째 배열에는 0
1번 째 배열에는 1
2번째 배열에는 2
3번째 배열에는 3을 넣고 있어요.

| 0 | 1 | 2 | 3 | 배열 반환!

배열 객체를 선언만 하고, 할당은 [14행]에서 해요
위 { 0, 1, 2, 3 } 배열을 할당

[15행]을 보면, 0부터 배열을 출력하는데
arr의 길이만큼 (전체)를 출력하도록 반복하고 있어요.
정답은 0123 이에요

[기출 예제] 22년 2회 실기 (난이도: ★☆☆)

▶ 정답 61

아래 자바 코드에서 출력되는 값을 작성하시오.

실제 실행코드

```
1  class Main {
2      public static void main(String[] args) {
3          ①Cond obj = new Cond(3);
4          ④obj.a=5;
5          ⑤int b = obj.func();
6          ⑦System.out.print(obj.a + b);
7      }
8  }
```

Cond 객체 생성 및 생성자에 3을 넘겨주고 있어요. Cond 안의 멤버변수 a에 3이 할당돼요.
하지만, [4행] 에서 바로 a의 값을 5로 덮어쓰고 있어요.
[16행]이 실행돼요. 56이 반환돼요. (로직은 아래에서 설명할게요)
obj.a + b 는 5 + 56이므로, 정답은 **61**이에요.

Cond 클래스

```
9
10 class Cond {
11     int a;
12  ② public Cond(int a) {
13         ③this.a = a;
14     }
15
16     ⑥public int func() {
17         int b = 1;
18         for (int i = 1; i < a; i++) {
19             b += a * i;
20         }
21         return a + b;
22     }
23 }
```

매개변수 a를 받아서 Cond 멤버변수 a에 할당해요. 여기서는 [3행]에 의해 3이 할당되었다가 [4행]에 의해 **5**가 되었어요.

b의 시작값 1이에요.

1 부터 a 미만까지 +1 증가하면서 반복해요.
여기서 a는 5여서, 1~4까지 반복해요.

a * i 는 (5 * 1) 한 값을 b에 더해줘요.
순서대로 를 반복해서 b에 더해줘요.
즉 b의 시작값 1 + (5 * 1) + (5 * 2) + (5 * 3) + (5 * 4) 이 b에 할당돼요.
b는 **51**이에요.
func() 메소드의 반환은 **a + b**이니, 5 + 51을 반환하면 **56**가 반환되요.

[기출 예제] 22년 3회 실기 (난이도: ★☆☆)

▶ 정답 996

아래 자바 코드에서 출력되는 값을 작성하시오.

실제 실행코드

```
1  class Exam{
2      public static void main(String[] args){
3          int a = 0;
4
5          for(int i = 1; i < 999; i++) {
6              if(i % 3 == 0 && i % 2 == 0)
7                  a = i;
8          }
9
10         System.out.print(a);
11     }
12 }
```

정수형 변수 a를 선언하고 0을 할당합니다.

1부터 998까지 (999미만) +1씩 증가하면서 반복해요.
%는 나머지를 뜻해요.

즉, **i를 3으로 나누고 남은 나머지 가 0**이면서, **i를 2로 나누고 남은 나머지가 0**인 경우에 a에 i를 할당해라 라는 의미에요.
이 말만 보면 **3의 배수**이면서 **2의 배수**인 **6의 배수**를 찾으라는 말인데요!
6의 배수마다 a에 계속에서 덮어쓰는 코드이니, 사실상 출력되는 값은 998에서 가장 가까운 6의 배수가 되겠네요!

998은 3의 배수가 아니고, 997은 2의 배수가 아닙니다. 996는 3의 배수 면서 2의 배수입니다!
그러므로, 정답은 996이 됩니다!

[기출 예제] 24년 3회 실기 (난이도: ★☆☆)

▶ 정답 00AAA

아래 자바 코드에서 출력되는 값을 작성 하시오.

```
1  public class Main {
2      static String[] s = new String[3];
3
4      static void func(String[] s, int size) {
5          for (int i = 1; i < size; i++) {
6              if (s[i - 1].equals(s[i])) {
7                  System.out.print("O");
8              } else {
9                  System.out.print("N");
10             }
11         }
12         for (String m : s) {
13             System.out.print(m);
14         }
15     }
16
17
18     public static void main(String[] args) {
19         s[0] = "A";
20         s[1] = "A";
21         s[2] = new String("A");
22
23         func(s, 3);
24     }
25 }
```

길이가 3인 문자열(String) 배열 s를 선언해요.

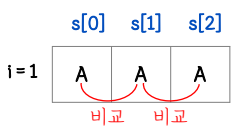
배열(s)과 정수(size)를 입력으로 받아 실행돼요.
1부터 size보다 작을 때까지 반복해요. (여기서는 size 3 [23행]이 들어가요.)
i 의 이전 값과 현재 값을 비교해요. (배열 주소가 아닌 값의 비교)
값이 같다면 O를 출력 (개행 없음)
값이 다르다면 N를 출력 (개행 없음)

배열(s)의 모든 값을 출력해요. (개행 없이 쭉 나열해서 출력해요)

실행 코드

배열(s)의 0 번째 위치에 문자열 A를 넣어요.
배열(s)의 1 번째 위치에 문자열 A를 넣어요.
배열(s)의 2 번째 위치에 문자열 A를 넣어요.

func 함수를 호출합니다.
위에서 설명했듯이 1 인덱스 부터 시작해서 2(3보다 작은)까지 수행해요.



s[0]와 s[1] 을 비교, s[1]과 s[2]를 비교해요.
전부 A 문자열이 들어가 있기에 OO를 출력해요.
이후, s 배열 전체를 출력해요. AAA가 출력돼요.

즉, 정답은 OOAAA가 돼요.

[기출 예제] 22년 3회 실기 (난이도: ★★☆)

아래 자바 코드에서 출력되는 값을 작성하시오.

실제 실행코드

```
1  public class Test {
2      public static void main(String[] args) {
3          int result[] = new int[5];
4
5          int arr[] = { 77, 32, 10, 99, 50 };
6
7          for (int i = 0; i < 5; i++) {
8              result[i] = 1;
9
10             for (int j = 0; j < 5; j++) {
11                 if(arr[i] < arr[j]) {
12                     result[i]++;
13                 }
14             }
15         }
16
17         for (int k = 0; k < 5; k++) {
18             System.out.print(result[k]);
19         }
20     }
21 }
```

▶ 정답 24513

n[0] n[1] n[2] n[3] n[4]

비어 있는 배열 5칸이 생성돼요.

크기가 5인 정수형 배열 result를 선언하고 생성해요.
배열은 같은 타입의 데이터를 연속적으로 저장하는 자료구조입니다.
배열의 크기는 생성 시에 결정되며, 이 경우 result 배열은 5개의 정수를 저장할 수 있어요

크기가 5인 정수형 배열 arr을 선언하고, 초기값으로 77, 32, 10, 99, 50을 저장합니다.

n[0]	n[1]	n[2]	n[3]	n[4]
77	32	10	99	50

배열 선언과 동시에 할당해요.
이걸 초기화 라고 불러요.

i 가 0부터 4까지 (5보다 미만) +1씩 증가하면서 반복해요.

j 가 0 부터 4까지 +1씩 증가하면서 반복해요.

	n[0]	n[1]	n[2]	n[3]	n[4]
i = 0	1				

	n[0]	n[1]	n[2]	n[3]	n[4]
i = 1	1	1			

이중 반복문으로
i 가 0일 때, j 가 0 ~ 4까지 반복하고,
i 가 1일 때, j 가 0 ~ 4까지 반복하고,
…
i 가 4일 때까지 반복하는 거에요.

그 안에 조건문 if 는 다음과 같아요.
arr의 i번째의 값이 arr의 j번째의 값보다 작으면,result[i]의 값을 +1 증가시켜요.
(result[i]의 값은 [8행]에서 넣어준 1 이 시작 값이에요.)

arr[0] 인 77보다 큰 값은 99 하나에요. 즉, 1을 더해줘요. 1 + 1 = **2**
arr[1] 인 32보다 큰 값은 77, 99, 50 세개 에요. 즉, 3을 더해줘요. 1 + 3 = **4**
arr[2] 인 10보다 큰 값은 77, 32, 99, 50 네개 에요. 즉, 4를 더해줘요. 1 + 4 = **5**
arr[3] 인 99보다 큰 값은 없어요. 즉, 그냥 **시작값 1**이에요.
arr[4] 인 50보다 큰 값은 77, 99 두개 에요. 즉, 2를 더해줘요. 1 + 2 = **3**
정답은 **24513** 이에요.
(한 줄 내리지 않아야 해요. print 는 줄 내림 없이 출력하고, println 은 출력 후 한 줄 내려요)

★ key point
이중으로 for문이 동작하는 것뿐, 어렵지 않아요!.
코드를 해석하는 게 어렵다면, **i = 0**을 한번 손으로 직접 계산해보세요. 규칙이 보일거에요.

▶ 정답 0+1+2+3+4+5=15
그냥 15만 적으면 틀려요!

[기출 예제] 21년 1회 실기 (난이도: ★★☆)

클래스 내에서 객체 생성 없이 사용할 수 있는 메소드로써 출력 결과를 작성하시오.

문제가 크게 어렵지는 않았지만, **조심해야할 포인트가 몇가지** 있어요!
1. println이 아닌 print라 줄내림없이 출력할 것!
2. 더하기 결과값인 15만 출력하는 것이 아닌, 0+1+…=15까지 전부 출력할 것!
실수하지 않도록 조심해야해요!

실제 실행코드

```
1  public class Main {
2      public static void main(String[] args) {
3          int i, j;
4          for(j=0, i=0; i<=5; i++) {
5              j+=i;
6              System.out.print(i);
7              if(i==5) {
8                  System.out.print("=");
9                  System.out.print(j);
10             } else {
11                 System.out.print("+");
12             }
13         }
14     }
15 }
```

j와 i에 0을 할당하고 (시작값), i가 5보다 작거나 같을 때까지 +1 증가하면서 반복해요

+= 연산자는 j = j + i 로 표현할 수 있어요. 즉, 자기자신과 i를 계속 더해줘요.

i가 0, 1, 2, 3, 4, 5까지 순서대로 출력돼요. println이 아닌 print이므로 줄내림하지 않아요.

i가 5일 때는 + 대신에 =를 출력하기 위한 조건문이에요.

i가 5일 때, =를 출력하고 여태 i를 계속 더했던 j를 출력해요.

i가 5가 아닌 경우에는 +를 출력하는 코드에요.

[문제 유형] 단순 while문

★ while

특정 조건이 참인 동안 계속해서 코드 블록을 실행하게 해요. 마치 "아직 배가 고프다면 계속 음식을 더 먹어"라고 말하는 것과 비슷해요.
조건이 거짓이 되는 순간, 반복이 멈춰요.

```
int i = 0; // 'i'라는 변수를 0으로 시작해요.
while (i < 5) { // 'i'가 5보다 작은 동안 계속 반복해요.
    System.out.println("i의 값은 " + i + "입니다."); // 현재 'i'의 값을 출력해요.
    i++; // 그리고 'i'를 1씩 증가시켜요.
}
```

for 문과의 차이

while 문은 조건이 참인 동안 계속 반복되는데, 주로 반복 횟수가 명확하지 않을 때 사용해요.
for 문은 반복 횟수가 정해져 있거나, 반복을 위한 변수의 시작과 끝이 명확할 때 주로 사용돼요.

★ continue

continue를 만나면, 반복문의 현재 반복을 즉시 종료하고 다음 반복으로 넘어가요.
마치 "이번 단계는 여기까지 하고, 다음 단계로 바로 넘어가자"라는 뜻이에요.
반복문의 나머지 부분을 건너뛰고, 반복문의 시작 부분으로 돌아가 새로운 반복을 시작해요.

★ break

break는 반복문을 완전히 끝내는 역할을 해요.
마치 "여기까지만 하고, 반복문을 완전히 벗어나자"라고 할 때 쓰는 것과 같아요.
break를 만나면, 그 즉시 반복문이 종료되고 반복문 바깥의 다음 코드로 넘어가요.

```
for (int i = 0; i < 10; i++) {
    if (i == 5) {
        continue; // i가 5일 때, 여기서 나머지 반복문을 건너뛰고 다음 반복으로 넘어가요.
    }
    if (i == 8) {
        break; // i가 8이 되면, for 문을 완전히 끝내요.
    }
}
```

[기출 예제] 20년 3회 실기 (난이도: ★☆☆)

▶ 정답 30

다음은 자바 코드이다. 출력 결과를 쓰시오.

실제 실행코드

```
1  public class Main {
2      public static void main(String[] args) {
3          int i=0;
4          int sum=0;
5          while (i<10) {
6              i++;
7              if(i%2 ==1)
8                  continue;
9              sum += i;
10         }
11         System.out.println(sum);
12     }
13 }
```

3행: i 가 0으로 선언 및 할당 되었어요.
4행: sum도 0으로 선언 및 할당 되었어요.
5행: i가 10 미만일 때까지 반복해요.여기서는 0,1,2,3,4,5,6,7,8,9 까지 반복해요
6행: i의 값을 바로 +1 증가시켜요. (while문 들어가면서 i가 아닌 i+1로 시작하니 주의해주세요!)
7행: i 를 2로 나눈 나머지가 1일 때, (의미상 홀수를 뜻해요)
8행: continue해요. 이 키워드는 다음 반복문으로 넘어간다는 의미에요.
여기서는 [5행]으로 올라가요. (즉, [9행]을 수행하지 않기 위함이에요)

[7행] 조건문에 걸리지 않았을 경우, (짝수일 때)
9행: sum 변수에 i를 더해요.
즉, 1,2,3,4,5,6,7,8,9,10 중에서 짝수만 더한 결과값을 출력하겠다는 코드에요

정답은 2+4+6+8+10=30이에요.

[문제 유형] 상속

★ 상속 (Inheritance)

상속은 부모 클래스가 가진 특성(변수)과 행동(메서드)을 자식 클래스가 물려받는 것을 말해요. 마치 부모님의 유전자를 물려받아 닮은 것처럼, 자바에서는 클래스도 다른 클래스의 특성을 물려받을 수 있어요.

```
class Parent {
    void sayHello() {
        System.out.println("Hello, I'm a parent!");
    }
}

class Child extends Parent {
    // Child 클래스는 Parent 클래스의 sayHello 메서드를 상속받았어요.
}
```

Child 클래스는 Parent 클래스로부터 sayHello 메서드를 상속받으니, Child 객체를 만들어 sayHello를 호출하면 "Hello, I'm a parent!"라는 메시지가 출력될 거예요.

★ 오버라이딩 (Overriding)

오버라이딩은 상속받은 메서드를 자식 클래스에서 재정의하는 것을 말해요.
자식 클래스가 상속받은 메서드를 마음에 들게 바꾸고 싶을 때 사용해요.

```
class Child extends Parent {
    void sayHello() {
        System.out.println("Hello, I'm a child!");
    }
}
```

여기서 Child 클래스는 Parent의 sayHello 메서드를 오버라이딩해서 "Hello, I'm a child!"라는 새 메시지를 출력하도록 변경했어요.

★ 오버로딩 (Overloading)

오버로딩은 같은 이름의 메서드를 여러 개 가지되,
매개변수의 유형이나 개수를 다르게 하는 것을 말해요.
이렇게 하면 같은 일을 하는 메서드라도 조금씩 다른 상황에서 사용할 수 있어요.

```
class MathOperations {
    int sum(int a, int b) {
        return a + b;
    }

    // 오버로딩된 sum 메서드, 매개변수의 개수가 다름.
    int sum(int a, int b, int c) {
        return a + b + c;
    }
}
```

MathOperations 클래스에는 sum이라는 메서드가 두 개 있어요. 하나는 두 숫자를 더하고, 다른 하나는 세 숫자를 더해요. 매개변수가 다르기 때문에 두 메서드는 서로 다른 버전으로 존재할 수 있어요.

위 **세개의 개념**은 자바에서 정말 중요해요! 먼저, 이해하시면 좋아요

[기출 예제] 23년 3회 실기 (난이도: ★★☆)

다음은 Java 코드이다. 올바른 출력 결과를 작성 하시오.

▶ 정답 "BDCDD"

실제 실행코드

```
1  public class main{
2      public static void main(String[] args) {
3      ① A b = new B();
4      ② b.paint();
5      ⑭ b.draw();
6      }
7  }
```

② b.paint(); → 자식 클래스 B의 paint( ) 실행 : [22행] → [14행] 실행 → [15행] **"B"출력** → [16행] → [26행] 실행 → [27행] **"D" 출력**
이어서 [22행] 다음의 [23행] 실행 **"C"출력** → [24행] 실행 → [26행] 실행 → [27행] **"D" 출력** (끝)

⑭ b.draw(); → 자식 클래스 B의 draw( ) 실행 : [26행] 실행 → [27행] **"D" 출력** (끝)

부모 클래스 (A)

```
9  class A {
10     public void paint() {
11         System.out.print("A");
12         draw();
13     }
14   ⑤ public void draw() {
15       ⑥ System.out.print("B");
16       ⑦ draw();
17     }
18 }
```

public void paint() { → 클래스 A의 메소드와 클래스 B의 메소드 이름이 동일 (= 자식 클래스인 B 메소드 실행)

⑦ draw(); → 자식 클래스 draw( ) = **[26행]**

(오버라이딩 : 동일한 이름의 메소드일 때 자식 클래스 메소드만 실행)

★ super.~ : 부모의 메소드를 호출
★ this.~ : 현재 실행 중인 나의 메소드 호출 / 생략도 가능 → draw() = this.draw()
super와 this에 대한 자세한 설명은 다음장에 있어요.

자식 클래스 (B)

```
19
20 class B extends A {
21   ③ public void paint() {
22       ④ super.draw();
23       ⑩ System.out.print("C");
24       ⑪ this.draw();
25     }
26 ⑧⑫⑮ public void draw() {
27   ⑨⑬⑯ System.out.print("D");
28     }
29 }
30
```

class B extends A { → 자식 클래스 B는 부모 클래스 A로부터 상속!

④ super.draw(); → 부모 클래스 A의 draw() 호출 = **[14행]**

⑪ this.draw(); → 자식 클래스 B의 draw() 호출 = **[26행]**

★ this 키워드

this는 현재 객체, 즉 메서드나 생성자를 호출하는 객체를 가리키는 데 사용돼요.
같은 이름의 클래스 변수와 메서드 매개변수가 충돌할 때 이를 구별하기 위해 사용되기도 해요.
클래스 내의 다른 생성자를 호출할 때도 this()를 사용할 수 있어요.

```
class Flower {
    private String name;

    Flower(String name) {
        this.name = name; // 여기서 'this.name'은 클래스 변수를, 'name'은 생성자의 매개변수를 가리켜요.
    }

    void printName() {
        System.out.println(this.name); // 'this'로 클래스 변수 'name'에 접근해요.
    }
}
```

여기서 this.name은 클래스의 name 변수를 참조하고, 단순히 name은 생성자로 전달된 매개변수를 참조해요.

★ super 키워드

super는 부모 클래스를 가리키는 데 사용되며, 부모 클래스의 변수나 메서드, 생성자에 접근할 때 사용해요.
자식 클래스에서 오버라이드한 메서드가 부모 클래스의 메서드를 호출하고 싶을 때 super를 사용해요.

```
class Plant {
    String type = "Plant";

    void printType() {
        System.out.println(this.type); // 'this'로 자신의 type 변수에 접근해요.
    }
}

class Flower extends Plant {
    String type = "Flower";

    void printType() {
        super.printType(); // 'super'로 부모 클래스의 printType 메서드를 호출해요.
        System.out.println(this.type); // 'this'로 자신의 type 변수에 접근해요.
    }
}
```

Flower 클래스에서 super.printType()은 Plant 클래스의 printType 메서드를 호출하고, this.type은 Flower 클래스 자신의 type 변수를 참조해요.

[기출 예제] 20년 2회 실기 (난이도: ★☆☆)

▶ 정답 a=10

다음은 자바 코드이다. 출력 결과를 쓰시오.

```
1  class A {
2      private int a;
3    ④public A(int a){
4          this.a = a;
5      }
6    ⑥public void display(){
7          System.out.println("a=" + a);
8      }
9  }
10
11 class B extends A {
12   ②public B(int a){
13       ③super(a);
14       ⑤super.display();
15     }
16 }
17
18
19 public class Main {
20     public static void main(String[] args){
21       ①B obj = new B(10);
22     }
23 }
```

A 부모 클래스

B 자식 클래스

실제 실행코드

a의 멤버 변수를 10을 할당해요.

super()는 부모 클래스의 생성자를 호출하는 키워드에요.
[3행]이 수행돼요.

부모의 [6행] display()를 호출해요.
즉, a에 10을 할당한 뒤, a를 출력하는 코드에요.
정답은 **a=10**이에요

B의 생성자에 10을 넣어 B 객체를 생성합니다.
[12행]이 수행돼요.

[기출 예제] 21년 2회 실기 (난이도: ★☆☆)

▶ 정답 11

다음은 JAVA 관한 문제이다. 알맞는 출력값을 작성하시오.

실제 실행코드

Ovr2 자식 클래스

```
1  public class Ovr1 {
2      public static void main(String[] args) {
3          ①Ovr1 a1 = new Ovr1();
4          ②Ovr2 a2 = new Ovr2();
5          ③System.out.println(a1.sun(3,2) + a2.sun(3,2));
6      }
7
8      ④int sun(int x, int y) {
9          ⑦return x + y;
10     }
11 }
12
13 class Ovr2 extends Ovr1 {
14
15     ⑤int sun(int x, int y) {
16         return x - y + super.sun(x,y);
17         ⑥
18     }
19 }
```

Ovr1 부모 클래스

①Ovr1 a1 = new Ovr1(); Ovr1 타입의 객체 a1을 생성하고 있어요.

②Ovr2 a2 = new Ovr2(); Ovr2 타입의 객체 a2를 생성하고 있어요. Ovr2는 Ovr1를 상속받았기 때문에 Ovr1의 메서드를 가지고 있어요.

③ → Ovr1의 sun은 3+2 = 5이고,
Ovr2의 sun은 3-2+5 = 6이기 때문에 정답은 11이 출력돼요.

④⑦ → Ovr1 클래스에 sun이라는 메서드를 정의하고 있어요.
두 정수를 받아서 그 합을 반환합니다.

⑤ → sun 메서드를 재정의(오버라이드)하고 있어요.
이 메서드는 인자로 받은 두 정수의 차에 부모 클래스의 sun 메서드의 결과를 더해 반환합니다.
super.sun(x,y)는 부모 클래스인 Ovr1의 sun 메서드를 호출하는 방법이에요.

★extends

class 자식 extends 부모 의 형태로 사용해요.
부모를 상속 받은 자식은 부모의 변수 및 메소드를 사용할 수 있고,
오버라이드를 통해 내 코드로 수정해서 사용할 수 있어요.

여기서는 [15행]이 부모의 sun을 오버라이드해서 사용하고 있어요.
이럴 경우, 자식 코드에서 부모의 메소드를 호출 하고 싶을 때는
super.메소드 명 으로 사용할 수 있어요.

[기출 예제] 23년 1회 실기 (난이도: ★★☆)

▶ 정답 5000

아래 자바 코드에서 출력되는 값을 작성 하시오.

부모 클래스

자식 클래스

실제 실행코드

```
1  class Parent {
2      int x = 1000;
3
4      Parent() {
5          this(3000);
6      }
7
8      Parent(int x) {
9          this.x = x;
10     }
11 }
12
13 class Child extends Parent {
14     int x = 4000;
15
16     ②Child() {
17         ③this(5000);
18     }
19
20     ④Child(int x) {
21         ⑤this.x = x;
22     }
23
24     ⑦int getX() {
25         ⑧return x;
26     }
27 }
28
29 public class Main {
30     public static void main(String[] args) {
31         ① Child obj = new Child();
32         ⑥ System.out.println(obj.getX());
33     }
34 }
```

기본 생성자 [4행]은 [8행] 생성자를 호출해서, X에 3000을 할당해요.

기본 생성자 [16행]은 [20행] 생성자를 호출해서, X에 5000을 할당해요.

만약, getX()가 (Child에 없고) Parent에만 구현되어 있었다면, 정답은 3000이 되었을거에요.
코드를 보면, getX()는 자기 클래스의 x를 반환하니까요.

Child를 기본 생성자로 생성했어요. [16행] -> [17행] -> [20행] -> [21행] 의 순서에 의해 Child의 x는 5000이 되었어요.
Child의 getX() 메소드는 [24행] Child의 x를 반환하기 때문에 정답은 **5000**이 돼요.

[기출 예제] 23년 3회 실기 (난이도: ★★☆)

▶ 정답 2

다음은 Java 코드이다. 올바른 출력 결과를 작성하시오.

부모 클래스 (Parent)

```
1  class Parent {
2      int compute(int num) {
3          if(num <= 1)
4              return num;
5          return compute(num-1) + compute(num-2);
6      }
7  }
```

이 문제에서는 호출되지 않아요.

[19행] 에서 Parent obj = new Parent(); 로 할당 되었다면 해당 메소드가 호출되었을 것.

자식 클래스 (Child)

```
9  class Child extends Parent {
10     int compute(int num) {
11         if(num <= 1)
12             return num;
13         return compute(num-1) + compute(num-3);
14     }
15 }
```

실제 호출되는 메소드.

n이 1 이하일 때, n을 반환한다 (종료 조건)

실제 실행코드

```
16
17 public class main {
18     public static void main(String args[]) {
19         Parent obj = new Child();
20         System.out.print(obj.compute(7));
21     }
22 }
23
```

자식 클래스 Child 객체를 부모 클래스 Parent에 할당.

껍대기는 부모 클래스 이지만, 자식 객체가 할당되었으므로,
compute 메소드는 자식클래스인 Child 객체의 메소드로 동작해요.
로직의 수행은 왼쪽 선언된 타입이 아닌 **오른쪽에 할당되는 객체의 메소드로 동작한 다는 사실**을 기억하면 좋아요!

자기 자신을 계속 호출하고 있어요.
[20행]에서 7을 넣어줬으니, compute(6) + compute(4) 가 호출돼요.
* compute(n) 에서 n이 1 이하일 때, n을 반환해요.
즉, compute(6) 의 값과 **compute(4)** 의 값을 구해서 더해주면 정답이에요.

compute(6) 은 compute(5) + compute(3) 의 합 — 1 + 0 = 1
compute(5) 는 compute(4) + compute(2) 의 합 — 1 + 0 = 1
compute(4) 는 compute(3) + compute(1) 의 합 — 0 + 1 = 1
compute(3) 은 compute(2) + compute(0) 의 합 — 0 + 0 = 0
compute(2) 는 compute(1) + compute(-1) 의 합 — 1 + (-1) = 0

compute(1) + compute(-1) 부터 거꾸로 계산하면서 올라가면서 풀어요.
즉, **compute(6) 는 1, compute(4) 는 1**이므로, **정답은 2**

▶ 정답 1

[기출 예제] 20년 4회 실기 (난이도: ★★☆)

다음은 자바 소스 코드이다. 출력 결과를 쓰시오.

Parent 부모 클래스

```
1  class Parent{
2      public int compute(int num){
3          if(num <=1) return num;
4          return compute(num-1) + compute(num-2);
5      }
6  }
```

Child 자식 클래스

```
8  class Child extends Parent {
9      public int compute(int num){
10         if(num<=1) return num;
11         return compute(num-1) + compute(num-3);
12     }
13 }
```

매개변수 num이 1과 같거나 작을 때, num을 반환해요. (종료조건)

실제 실행코드

```
15 class Main{
16     public static void main (String[] args){
17         Parent obj = new Child();
18         System.out.print(obj.compute(4));
19     }
20 }
```

[9행]을 실행시켜요.

자기 자신을 계속 호출하고 있어요.
[18행]에서 4를 넣어줬으니, compute(3) + compute(1) 이 호출돼요.
- compute(n) 에서 n이 1 이하일 때, n을 반환해요. (종료조건)

즉, compute(3) 의 값과 **compute(1)** 의 값을 구해서 더해주면 정답이에요.

compute(3) 은 compute(2) + compute(0) 의 합 — 0 + 0 = 0
compute(2) 는 compute(1) + compute(-1) 의 합 — 1 + (-1) = 0
compute(1) 은 1과 같으니 ([10행] 조건문), 1을 반환해요. = 1
즉, **compute(3) 는 0, compute(1) 는 1**이므로, **정답은 1 이에요**

Parent 부모 클래스를 선언 해 실제 객체는 자식인 Child 를 생성해 할당 했어요.

동일한 메소드가 부모, 자식에 존재할 경우, 자식 클래스의 메소드를 호출되고,
- 위의 경우를 오버라이드 라고 표현해요. (부모의 메소드를 자식이 새롭게 구현하는 경우)
- 단, 메소드 이름, 매개변수들의 개수와 타입, 반환 타입이 전부 같아야 해요

부모에만 존재하는 메소드일 경우에만 부모 메소드로 호출돼요.

[18행] compute메소드의 경우, 자식에도 동일한 메소드가 존재하므로,
자식의 compute가 호출되고, 부모 compute는 무시됩니다.

23년도 3회 실기 문제와 숫자만 다른 동일한 문제에요.

[기출 예제] 20년 2회 실기 (난이도: ★☆☆)

▶ 정답 new

다음은 자바 코드이다. 다음 밑줄에 들어갈 키워드를 쓰시오.

Parent 부모 클래스

Child 자식 클래스

실제 실행코드

```
1  class Parent {
2      public void show() {
3          system.out.println("Parent");
4      }
5  }
6
7  class Child extends Parent {
8     ③public void show() {
9         ④system.out.println("Child");
10     }
11 }
12
13 public class Main {
14     public static void main(String[] args) {
15        ①Parent pa = _____ Child();
16        ②pa.show();
17     }
18 }
```

Parent 부모 클래스의 show()가 Child 자식 클래스 show로 오버라이드 되었어요.

즉, [16행] 호출 시, [8행]의 show가 호출돼요.
물론, 이 문제와는 상관없어요!

★ key point
이제는 아마, 나올 일 없는
가장 기본적인 자바의 객체 생성에 대한 문법 문제이에요.

[8행]이 호출돼요.

자바에서 생성자를 만드는 기본 문법은 A a = new A()입니다
A는 클래스명이에요.
여기서는 Parent pa = new Child(); 이 부분이에요.
그래서 정답은 new가 돼요

가장 기본적인 문법이에요.

[기출 예제] 24년 1회 실기 (난이도: ★☆☆)

해당 문제는 **실행 순서만을 작성**하는 것이기 때문에 절대 어렵지 않아요.
코드 파라미터 등을 신경 쓰지 말고 순서대로 따라가면 돼요.

▶ 정답
6 -> 3 -> 1 -> 7 -> 2

아래 JAVA언어 코드의 실행 순서를 중복 번호없이 작성하시오.
실행 순서 : 5 → ( ) → ( ) → ( ) → ( ) → ( )

```
1  class Parent {
2      int x, y;
3
4      Parent(int x, int y) { ①
5          this.x=x;
6          this y=y;
7      }
8
9      int getT() { ②
10         return x*y;
11     }
12 }
13
14 class Child extend Parent {
15     int x;
16
17     Child (int x) { ③
18         super(x+1, x);
19         this.x=x;
20     }
21
22     int getT(int n){ ④
23         return super.getT()+n;
24     }
25 }
26
27 class Main {
28     public static void main(String[] args) { ⑤
29         Parent parent = new Child(3); ⑥
30         System.out.println(parent.getT()); ⑦
31     }
32 }
```

우리는 main 부터 로직이 수행되는 걸 알고 있습니다.
(모르고 있더라도 보기에서 5번이라고 적어줬기 때문에 상관없습니다)
5번 부터 수행합니다.

로직은 한 줄 씩 수행하면서 내려가기 때문에 6번이 수행됩니다.
new Child(3);이 수행되면서 Child 클래스의 생성자 3번을 호출합니다.
(1번이 아니라 3번이 호출되는 이유는 new Child(); 이기 때문이에요. 해당 라인에서 1번이 호출되려면 new Parent(); 가 되어야 해요.)

한 줄 내려가면 super 메소드를 호출합니다.
super는 부모 this 는 자기 자신인데, super() 는 super의 생성자를 뜻해요.
1번으로 이동해 super() 생성자를 수행하고, 돌아오면 [29행]이 종료돼요.

한 줄 내려가 7번이 수행되고,
parent.getT()가 수행되어야 출력될 수 있으니 2번이 수행되며 값을 반환하고
(4번이 아니라 2번일까? 하고 헷갈릴 수 있어요 .
Child는 Parent를 상속받고 있고 부모의 메소드를 전부 사용할 수 있어요. 2번과 4번은 메소드명만 getT로 같고 파라미터가 다르기 때문에 오버로딩으로 아예 다른 메소드에요.
parent.getT();는 파라미터가 없는 메소드 호출했기 때문에 Parent의 getT()가 호출된거에요.)

그 값이 출력되며 프로그램이 종료됩니다.

[기출 예제] 24년 1회 실기 (난이도: ★★☆)

▶ 정답 9

다음 Java 코드를 보고 알맞는 출력 값을 작성하시오.

Parent 부모 클래스

Child 자식 클래스

실제 실행코드

```
1  class classOne {
2      int a, b;
3
4      public classOne(int a, int b) {
5          this.a = a;
6          this.b = b;
7      }
8
9      public void print() {
10         System.out.println(a + b);
11     }
12
13 }
14
15 class classTwo extends classOne {
16     int po = 3;
17
18     public classTwo(int i) {
19         super(i, i+1);
20     }
21
22     public void print() {
23         System.out.println(po*po);
24     }
25 }
26
27 public class Main {
28     public static void main(String[] args) {
29         classOne one = new classTwo(10);
30         one.print();
31     }
32 }
```

★key point

a와 b에 값을 할당하는 로직을 수행해서,
Print 메소드도 부모 메소드로 호출되나? 하고 헷갈릴 수 있어요.

하지만, 부모 자식 간의 호출 방식을 이해하고 있었다면
정말 쉽게 풀 수 있는 문제에요

new classTwo(10);를 통해 객체를 생성합니다.
[18행]이 호출되고, [19행]에서 super(10, 11); 이 호출됩니다.
[4행]이 호출되고, a는 10, b는 11이 할당됩니다.

one.print();이 호출될 때 코드는 먼저 자식에 print()가 존재하는지 확인하고,
없으면 부모 print()를 수행해요.

자식에 print()가 존재하고 그 로직은 po 변수와 po변수를 곱한 값을 출력하는 거에요
정답은 3 * 3인 9가 돼요.

★key point

자바의 상속과 그에 따른 다형성과 멤버 변수 참조에 대해서 잘 아는지
체크하는 문제에요.

여기서 중요한 점은

- **멤버 변수**는 **참조 변수의 타입**을 따르고,
- **함수**는 **실제 객체의 타입**을 따라요.

[기출 예제] 24년 3회 실기 (난이도: ★★☆)

▶ 정답 52

다음은 Java 코드에 대한 문제이다. 아래 코드를 확인하여 알맞는 출력값을 작성하시오.

실제 실행 함수

부모 클래스

자식 클래스

```
1  public class Main {
2      public static void main(String[] args) {
3          Base a = new Derivate();
4          Derivate b = new Derivate();
5
6          System.out.print(a.getX() + a.x + b.getX() + b.x);
7      }
8  }
9
10
11 class Base {
12     int x = 3;
13
14     int getX() {
15         return x * 2;
16     }
17 }
18
19 class Derivate extends Base {
20     int x = 7;
21
22     int getX() {
23         return x * 3;
24     }
25 }
```

a라는 변수를 선언해 부모 타입 Base로 객체를 생성하지만,
실제로 자식 클래스 Derivate의 객체를 할당해요.
즉, a.x 는 3 이지만, a.getX()는 Derivate의 getX()를 호출해 7 * 3 = 21 이 돼요.

b는 자식 클래스 Derivate로 객체 생성 및 할당을 해요.
즉, b.x는 7이고, b.getX()도 7 * 3= 21 이 돼요.

그래서 21(a.getX()) + 3(a.x) + 21(b.getX()) + 7(b.x) = 52가 출력돼요.

[기출 예제] 25년 1회 실기 (난이도: ★☆☆)

다음은 Java 코드에 대한 문제이다. 아래 코드를 확인하여 알맞는 출력값을 작성하시오.

▶ 정답 54

실제 실행코드

```java
public class Main {
    public static void main(String[] args) {
        new Child();
        System.out.println(Parent.total);
    }
}
```

Parent 부모 클래스

```java
class Parent {
    static int total = 0;
    int v = 1;

    public Parent() {
        total += (++v);
        show();
    }

    public void show() {
        total += total;
    }
}
```

Child 자식 클래스

```java
class Child extends Parent {
    int v = 10;

    public Child() {
        v += 2;
        total += v++;
        show();
    }

    @Override
    public void show() {
        total += total * 2;
    }
}
```

★key point

아래 포인트를 알면 너무 쉬운 문제에요.

1. 생성자는 항상 **부모→자식** 순으로 실행돼요.
2. show()는 **오버라이딩된 자식 메서드가 호출**돼요.
3. static 변수 total은 **클래스 전체에서 하나만 공유**돼요.

- [3행] new Child()를 호출하면,
  1. Parent() 생성자가 호출
     - ① ++v = v (Parent) 1 → 2, **total = 0 + 2(++v) = 2**
     - ② show() 호출 → 오버라이딩된 child.show() 호출
       - total = **2(total) + (2(total) * 2) = 6**
  2. Child() 생성자 실행
     - ① v = **10(v) + 2 = 12**
     - ② total = **6 + 12(v) = 18 (그 뒤에 v는 13이 돼요)**
     - ③ show() 호출 → 오버라이딩된 child.show() 호출
       - total = **18 + (18*2) = 54**
- [4행] System.out.println(Parent.total)
  - 최종 total 값인 54가 출력돼요.

[문제 유형] 추상 클래스

★ abstract

자바에서 추상 클래스란, 완전하지 않은 '미완성 설계도' 같은 것이에요. 직접적으로 객체를 만들 수는 없지만, 다른 클래스들이 상속받아 완성해야 하는 기본 틀을 제공해요. 이 추상 클래스는 추상 메서드를 포함할 수 있어요. 추상 메서드는 실행할 구체적인 코드가 없고, 그저 메서드의 이름, 입력값, 반환 타입만 정의돼 있죠. 상속받는 클래스는 이 추상 메서드들을 반드시 구현(완성)해야 해요.

예를 들어, 모든 차에는 주행 기능이 있지만, 각기 다른 차의 주행 방식은 조금씩 다를 수 있어요. 이 때 '차'를 추상 클래스로 생각하고 '주행' 기능을 추상 메서드로 두면, 각 차는 자신의 방식대로 '주행' 메서드를 구현하게 되죠.

```
abstract class Car {
    abstract void drive(); // 추상 메서드: '어떻게 주행할지'는 여기서 정의하지 않아요.

    void startEngine() {
        System.out.println("엔진이 켜졌어요!"); // 일반 메서드: 모든 차가 공통적으로 사용해요.
    }
}

class Sedan extends Car {
    void drive() {
        System.out.println("매끄럽게 도로를 주행해요!"); // '주행'에 대한 구체적인 구현이 여기에 와요.
    }
}

class Truck extends Car {
    void drive() {
        System.out.println("강력하게 짐을 실고 달려요!"); // 트럭에 맞게 '주행'을 다르게 구현해요.
    }
}
```

여기서 Car는 추상 클래스로서, drive 메서드를 추상 메서드로 가지고 있어요. Sedan과 Truck은 Car 클래스를 상속받아 drive 메서드를 각자의 방식으로 구현하죠.
Car 클래스는 직접 new Car()로 객체를 만들 수 없지만, Sedan과 Truck은 실제 사용 가능한 차들이 되어요.

[기출 예제] 23년 1회 실기 (난이도: ★★☆)

다음 JAVA 코드에서 알맞는 출력 값을 작성하시오.

▶ 정답 Vehicle name: Spark

```
1  abstract class Vehicle {
2      ⑤String name;
3      abstract public String getName(String val);
4
5      ⑦public String getName() {
6              return "Vehicle name: " + name;
7      }
8  }
9
10
11
12 class Car extends Vehicle {
13     ④private String name;
14     ②public Car(String val) {
15         ③name=super.name=val;
16     }
17
18     public String getName(String val) {
19             return "Car name:" + val;
20     }
21
22     public String getName(byte val[]) {
23             return "Car name:" + val;
24     }
25 }
26
27
28 public class Main {
29
30     public static void main(String[] args) {
31
32         ①Vehicle obj = new Car("Spark");
33         ⑥System.out.println(obj.getName());
34
35     }
36 }
```

부모 클래스

자식 클래스

실제 실행코드

class 앞에 abstract 키워드가 붙으면 추상 클래스가 돼요.
이는 단독으로 구현(생성)될 수 없음을 의미해요.
여기서의 Car 클래스처럼 상속으로만 생성 가능해요.

abstract 키워드를 붙여 getName(val) 메소드의 구현을
명시적으로 자식에게 넘겼어요. (추상 클래스에서는 구현을 자식에게 미룰 수 있는 효과가 있어요)

getName() 메소드의 구현은 Vehicle에 되어있어요.
추상 클래스라고 모든 메소드를 구현하지 못하는 건 아니에요.

Car의 name 변수와 부모인 Vehicle의 name 변수를
동시에 val 매개변수로 할당하는 코드에요.

Vehicle의 추상 메소드 getName(String val)을 오버라이드해서
구현한 메소드에요. 여기서는 사용되지 않아요.

Vehicle 과 상관없는 Car 의 메소드에요.
여기서는 사용되지 않아요.

Vehicle 클래스에 Car 객체를 생성 후 할당했어요.
여기서는 [5행] Vehicle의 getName() 메소드가 호출돼요.
자식인 Car 객체에는 getName() 메소드가 없기 때문이에요

★ abstract
'완성(구현)되지 않은' 클래스나 메서드를 지정할 때 사용되며, 이를 상속받은 클래스에서 완성해야 합니다.

★ 오버라이드
상속받은 클래스에서 부모 클래스의 메서드를 재정의하여 새로운 기능으로 바꾸는 것이에요.

★ 오버로딩
같은 이름의 메서드를 매개변수만 다르게 해서 여러 개 만드는 것이에요.
이름이 같더라도, 매개변수의 타입이나 매개변수의 개수 가 다르면 다른 메소드처럼 사용할 수 있어요.

▶ 정답 Vehicle name: Spark

[기출 예제] 20년 3회 실기 (난이도: ★★☆)

다음은 자바 코드이다. 출력 결과를 쓰시오.

class 앞에 abstract 키워드가 붙으면 추상 클래스가 돼요.
이는 단독으로 구현(생성)될 수 없음을 의미해요.
여기서의 Car 클래스처럼 상속으로만 생성 가능해요.

Vehicle 부모 클래스

Car 자식 클래스

실제 실행코드

```
1  abstract class Vehicle{
2      private String name;
3      abstract public String getName(String val);
4
5      public String getName(){
6          return "Vehicle name:" + name;
7      }
8      public void setName(String val){
9          name = val;
10     }
11 }
12
13 class Car extends Vehicle{
14     public Car(String val){
15         setName(val);
16     }
17     public String getName(String val){
18         return "Car name : " + val;
19     }
20     public String getName(byte val[]){
21         return "Car name : " + val;
22     }
23 }
24
25 public class Main {
26     public static void main(String[] args){
27         Vehicle obj = new Car("Spark");
28         System.out.print(obj.getName());
29     }
30 }
```

abstract 키워드를 붙여 getName(val) 메소드의 구현을 명시적으로 자식에게 넘겼어요.
(추상 클래스에서는 구현을 자식에게 미룰 수 있는 효과가 있어요)

abstract 클래스도 메소드를 구현할 수 있어요.
구현하지 않고, 자식에게 구현을 미룰 수도 있구요.
여기서는 [28행] getName()은 Car(자식 클래스)에서 오버라이드 하지 않고 있어서, 부모의 [5행] getName()이 호출돼요.

즉, 정답은 Vehicle name: Spark 이에요.

사실상, Car 자식 클래스가 이 코드에서는 사용되지 않아요.

코드 설명만 해보자면, Vehicle의 [3행] 추상 메소드는 자식에서 무조건 구현해줘야 해요.
그게 [17행]에서 구현되었어요.

[20행]은 메소드명은 같지만, 매개변수 타입이 다르므로 전혀 다른 메소드에요. 그냥 Car에서만 구현된 Car만의 메소드에요.

[14행]은 Car의 생성자로, 매개변수 val를 부모인 Vehicle의 setName 메소드를 이용해서 name 변수에 값을 할당해주고 있어요.
(여기서는 SparK로 할당)

부모의 메소드는 super.setName(val)의 형태로 호출해야하지만,
따로 자식에서 동일한 메소드가 존재하지 않으면 super 키워드 없이도 호출 가능해요.
super를 붙여야 할 때는, 자식에도 setName 메소드를 오버라이딩 했을 경우에 부모의 setName을 호출하고 싶을 때 에요.

23년도 1회 실기 문제와 동일한 문제에요.

[문제 유형] 인터페이스

★ interface
추상클래스와 정말 유사하게 '구현되지 않은' 메서드를 미리 선언하고, 이를 상속받은 클래스에서 완성해야 합니다.

사실, 여기서는 interface의 개념을 모르더라도 크게 문제 풀이 지장이 없어요.
그냥 추상 클래스와 비슷하게 인터페이스를 상속받은 클래스는 인터페이스가 가진 메소드를 구현해야 한다는 개념만 알고 넘어가도 돼요.

추상 클래스와 차이점을 가볍게 설명하자면, (이해 하지 않아도 돼요)
추상 클래스는 멤버 변수를 가질 수 있고, 인터페이스는 멤버 변수를 가질 수 없어요.

[기출 예제] 24년 2회 실기 (난이도: ★★☆)

다음은 Java 언어에 대한 문제이다. 아래 코드를 확인하여 알맞는 출력값을 작성하시오.

▶ 정답
25, 20

실제 실행코드

```
1  class Main {
2      public static void main(String[] args) {
3          int a[] = {1, 2, 3, 4, 5, 6, 7, 8, 9};
4          ODDNumber OE = new ODDNumber();
5          System.out.print(OE.sum(a, true) + ", " + OE.sum(a, false));
6      }
7  }
```

- 3: 배열 a에 [1, 2, 3, 4, 5, 6, 7, 8, 9] 를 할당합니다.
- 4: ODDNumber 클래스를 생성 후 OE 변수에 할당해요.
- 5: OE.sum(a, true)와 OE.sum(a, false)의 값을 받아서 출력합니다.
  - sum의 인자로 true일 때, 배열 a의 홀수들의 합
  - false일 때, 짝수들의 합을 반환해요.
  - 모든 홀수의 합 (1 + 3 + 5 + 7 + 9 = 25)
  - 모든 짝수의 합 (2 + 4 + 6 + 8 = 20)
  - 그러므로 정답은 25, 20가 출력이 돼요

인터페이스

```
8
9  interface Number {
10     int sum(int[] a, boolean odd);
11 }
```

Number를 상속 받았기 때문에 메소드 sum을 구현해야 해요.

구현체

```
12
13 class ODDNumber implements Number {
14     public int sum(int[] a, boolean odd) {
15         int result = 0;
16         for(int i=0; i < a.length; i++) {
17             if((odd && a[i] % 2 != 0) || (!odd && a[i] % 2 == 0)) {
18                 result += a[i];
19             }
20         }
21         return result;
22     }
23 }
```

- 16: 배열 a의 0부터 끝까지 루프를 돌면서 아래의 조건에 해당 하는지 체크해요.
- 17: *(odd && a[i] % 2 != 0)*
  - odd가 true이면서 i 번째 배열의 값이 홀수 일 때
  - (a[i]를 2로 나눈 나머지가 0이 아닐 때)
  - 이거나 ||
  - *(!odd && a[i] % 2 == 0)*
  - odd가 false이면서 i 번째 배열의 값이 짝수 일 때
  - (a[i]를 2로 나눈 나머지가 0일 때)
- 18: result에 i번째 값을 더해줘요.
- 21: 배열을 다 돌면 result를 반환해요.

즉, odd가 true이면 배열의 홀수 값의 합
odd가 false이면 배열의 짝수 값의 합을
반환하는 메소드에요.

[문제 유형] JAVA 문법

[기출 예제] 21년 3회 실기 (난이도: ★☆☆)

▶ 정답 3

다음 Java 코드에 대한 알맞는 출력값을 쓰시오.

Connection 클래스 (싱글톤 패턴)

```
1  class Connection {
2      private static Connection _inst = null;
3      private int count = 0;
4
5      static public Connection get() {
6          if (_inst == null) {
7              _inst = new Connection();
8              return _inst;
9          }
10         return _inst;
11     }
12
13     public void count() {
14         count++;
15     }
16
17     public int getCount() {
18         return count;
19     }
20 }
21
22 public class testcon {
23     public static void main(String[] args) {
24         Connection conn1 = Connection.get();
25         conn1.count();
26         Connection conn2 = Connection.get();
27         conn2.count();
28         Connection conn3 = Connection.get();
29         conn3.count();
30
31         System.out.print(conn1.getCount());
32     }
33 }
```

- 2행: _inst라는 정적 변수를 사용해서 인스턴스를 저장하고 있어요. 처음에는 _inst가 null이에요.
- 5~11행: get() 메서드를 호출하면 _inst가 null인지 확인해요. 만약 null이면 새로운 Connection 인스턴스를 생성하고 _inst에 할당해요. 이미 _inst에 인스턴스가 있다면 그 인스턴스를 그대로 반환해요.
  매번 새롭게 Connection을 생성하지 않고, 모두가 같은 객체를 바라보니 메모리 효율이 좋아져요.
- 13~15행: 호출 시, count 멤버 변수의 값을 +1씩 증가 시켜요.
- 17~19행: 호출 시, count 를 반환해요.

실제 실행코드

즉, [24행] [26행] [28행] 모두 같은 Connection 객체를 반환해요.
최초에 null일 경우에만 생성하고, 그 이후에는 이미 생성된 객체를 반환하니까요.

그러므로, Connection 객체 안의 멤버변수 count도 동일한 변수이고,
어디서 count() 메소드를 호출하더라도 같은 변수의 값이 +1씩 증가해요.
count()가 총 3번 호출되었으니, 정답은 **3**이에요.

★ 싱글톤 디자인 패턴

클래스의 인스턴스를 오직 하나만 생성하고, 그 인스턴스를 어디서든 접근할 수 있게 해주는 디자인 패턴이에요.

클래스의 인스턴스를 하나만 생성하고 싶을 때 사용해요.

인스턴스는 오직 하나만 생성되고, 프로그램 어디에서든 Connection.get()을 호출하면 같은 인스턴스를 받을 수 있어요.

[기출 예제] 24년 1회 실기 (난이도: ★☆☆)

21년도 3회 실기에서 나왔던 문제가 24년도에 동일하게 출제되었어요.
싱글톤 패턴과 static에 대해서 알고 있었다면 쉽게 맞출 수 있었어요.

▶ 정답 4

다음 Java 코드에 대한 알맞는 출력값을 쓰시오.

Connection 클래스 (싱글톤 패턴)

```
1  class Connection {
2
3      private static Connection _inst = null;
4      private int count = 0;
5
6      static public Connection get() {
7          if(_inst == null) {
8              _inst = new Connection();
9              return _inst;
10         }
11         return _inst;
12     }
13
14     public void count() {
15         count++;
16     }
17
18     public int getCount() {
19         return count;
20     }
21 }
22
23
24 public class main {
25
26     public static void main(String[] args) {
27
28         Connection conn1 = Connection.get();
29         conn1.count();
30
31         Connection conn2 = Connection.get();
32         conn2.count();
33
34         Connection conn3 = Connection.get();
35         conn3.count();
36
37         conn1.count();
38         System.out.print(conn1.getCount());
39     }
40 }
```

- 3행: _inst라는 정적 변수를 사용해서 인스턴스를 저장하고 있어요. 처음에는 _inst가 null이에요.
- 4행: count는 멤버 변수로 정적 변수는 아니지만 (= Connection 객체마다 생성되는 변수지만) Connection 객체가 _inst 변수로 싱글톤 패턴을 통해 한번만 생성되어 사용되기 때문에 사실상, count 변수는 하나의 객체에 하나의 변수로만 사용돼요.
- 6~12행: get() 메서드를 호출하면 _inst가 null인지 확인해요. 만약 null이면 새로운 Connection 인스턴스를 생성하고 _inst에 할당해요. 이미 _inst에 인스턴스가 있다면 그 인스턴스를 그대로 반환해요.
- 14~16행: 호출 시, count 멤버 변수의 값을 +1씩 증가 시켜요.
- 18~20행: 호출 시, count 를 반환해요.

실제 실행코드

[28행] [31행] [34행] 모두 같은 Connection 객체를 반환해요.

그러므로, Connection 객체 안의 멤버변수 count도 동일한 변수이고,
어디서 count() 메소드를 호출하더라도 같은 변수의 값이 +1씩 증가해요.
count()가 [29행] [32행] [35행] [37행] 총 4번 호출되었으니, 정답은 **4**이에요.

[문제 유형] JAVA 문법

▶ 정답 -8

[기출 예제] 22년 2회 실기 (난이도: ★☆☆)

아래 자바 코드에서 출력되는 값을 작성하시오.

실제 실행코드

```
1  class Test{
2      public static void main(String args[]){
3
4          int i = 3;
5          int k = 1;
6
7          switch(i) {
8              case 1: k += 1;
9              case 2: k++;
10             case 3: k = 0;
11             case 4: k += 3;
12             case 5: k -= 10;
13             default: k--;
14         }
15
16         System.out.print(k);
17     }
18 }
```

정수형 변수 i를 선언하고 3을 할당합니다.
정수형 변수 k를 선언하고 1을 할당합니다.

i 가 3, case 1은 통과해요.
i 가 3, case 2은 통과해요.
i 가 3, case도 3이기에, k에 0을 할당해요.
break; 가 없기 때문에 빠져나오지 못하고
case 4, case 5, default까지 로직을 수행해요.

즉, +3 (case 4) - 10 (case 5) -1 (default)를
전부 수행해서 k는 -8이 됩니다.

정답은 **-8**입니다.

★ switch

여러 개의 선택 사항 중 하나를 선택하여 실행하는
제어문이에요.

기본 구조는 다음과 같아요.

```
switch (변수) {
    case 값1:
        // 변수가 값1과 같을 때
        break;
    case 값2:
        // 변수가 값2와 같을 때
        break;
    ...
    default:
        // 어떤 case에도 해당하지 않을 때
        break;
}
```

break는 현재 case를 종료하고 switch 구문을 빠져나가는 키워드에요.
일치하는 case까지만 실행하고, switch 구문이 종료돼요.

여기서는 break 구문이 없으니, 빠져나오지 못해요.
그래서 [10행] **case 3에 걸린 이후로 아래의 case 로직들도 전부 수행돼요**

[기출 예제] 24년 2회 실기 (난이도: ★☆☆)

다음은 Java 코드에 대한 문제이다.
아래 코드를 확인하여 알맞는 출력값을 작성하시오.

▶ 정답
NNN

실제 실행코드

```
1  class Main {
2      public static void main(String[] args) {
3          int[] a = new int[] {1, 2, 3, 4};
4          int[] b = new int[] {1, 2, 3, 4};
5          int[] c = new int[] {1, 2, 3};
6
7          check(a, b);
8          check(a, c);
9          check(b, c);
10     }
11
12     public static void check(int[] a, int[] b) {
13         if (a == b) {
14             System.out.print("O");
15         } else {
16             System.out.print("N");
17         }
18     }
19 }
```

a: [1, 2, 3, 4]
b: [1, 2, 3, 4]
c: [1, 2, 3]
세 개의 숫자 배열을 생성 및 값을 할당합니다.

아래의 check 메소드에 위에서 생성한 배열들을 조합해 호출합니다.
인자로 넘겨진 두 배열 같은 객체이면 O, 다른 객체이면 N을 출력합니다.
(정확하게 표현하면, 객체의 주소를 비교합니다)
배열 안에 들어간 값과 전혀 상관없이 a, b, c 배열 모두 새롭게 별도로 생성된 객체이기에 별도의 주소를 가지고 있습니다.
그러므로 정답은 NNN이 됩니다.

★ 비교 연산자

개념만 알고 있다면 정말 쉽게 풀 수 있는 문제에요.
기본형 타입 (int, Boolean 등)의 경우, 그 값이 같은 지 비교합니다.
ex) int a = 1; int b = 1 일 때, a와 b는 같습니다.

객체 타입 (배열, String, 사용자 정의 클래스 등)의 경우, == 연산자로 비교할 때 메모리 주소를 비교합니다.
홍길동이라는 사람이 2명을 생성한다고 했을 때 그 둘이 같지 않은 것과 동일합니다.
ex) Person p1 = new Person("홍길동"); Person p2 = new Person("홍길동");
p1 == p2는 false가 됩니다.
메모리 주소 안에서 두 개의 객체가 생성되었고, 그 두 객체 주소는 다르니까요.

▶ 정답 static

[기출 예제] 21년 2회 실기 (난이도: ★☆☆)

클래스 내에서 객체 생성 없이 사용할 수 있는 메소드로써 빈칸에 들어갈 내용을 작성하시오.
[출력결과]
positive

실제 실행코드

```
1  public class Test {
2      public static void main(String[] args) {
3          System.out.print(Test.check(1));
4      }
5
6      (_____)  String check (int num) {
7          return (num >= 0) ? "positive" : "negative";
8      }
9  }
```

[3행]을 보면 객체 생성없이 메소드를 호출하고 있죠.
이런 경우는 **static** 키워드가 붙어있어야 가능해요.
물론, System.out.println(check(1)); 도 호출 가능해요.

★static
자바 프로그래밍에서 'static'을 사용하면, 그 요소는 특정 객체에 속하지 않고, 해당 클래스의 모든 객체가 공유합니다.

서로 다른 객체라도 static으로 선언된 변수는 그 클래스의 모든 인스턴스에서 공유되어, 어떤 객체가 그 값을 변경하면 다른 모든 객체에서도 그 변경된 값을 볼 수 있어요.

'static' 메소드도 클래스의 특정 인스턴스가 아니라 클래스 자체에 속해 있기 때문에, 객체를 생성하지 않고도 호출할 수 있습니다. 이런 특성 때문에 'static'은 클래스 수준에서 공유되는 변수나 메소드를 만들 때 유용하게 사용됩니다.

★만약, static을 안 쓰면서 호출하려면 어떻게 코드가 작성되어야 할까요?
[3행]이 아래와 같이 대체되면 동일하게 동작해요!
Test test = new Test();
System.out.print(test.check(1));

[기출 예제] 21년 1회 실기 (난이도: ★☆☆)

다음은 JAVA 관한 문제이다. 알맞는 출력값을 작성하시오.

▶ 정답
[4행] 3
[5행] 1
[6행] 45
[7행] 50
[8행] 89

	[0][0]	[0][1]	[0][2]
[0][0]	45	50	75
[1][0]	89		

```
1  public class Good {
2      public static void main(String[] args) {
3          int[][]arr = new int[][]{{45,50,75},{89}};
4          System.out.println(arr[0].length);
5          System.out.println(arr[1].length);
6          System.out.println(arr[0][0]);
7          System.out.println(arr[0][1]);
8          System.out.println(arr[1][0]);
9      }
10 }
```

3행 → 이차원 배열을 생성합니다.
4행 → 배열의 0번째에는 크기 3의 배열이 들어가 길이가 3이에요.
5행 → 배열의 0번째에는 크기 1의 배열이 들어가 길이가 1이에요.
6행 → 배열의 0번째의 0번째 값은 45에요.
7행 → 배열의 0번째의 1번째 값은 50에요.
8행 → 배열의 1번째의 0번째 값은 89에요.
첫 번째 배열에는 괄호의 가장 밖 값들이 들어가고, {45, 50, 75} | {89}
두 번째 배열부터는 그 값안에서의 배열이라고 생각하면 쉬워요.

그러므로 정답은 3, 1, 45, 50, 89의 값이 줄내림으로 출력돼요.

[기출 예제] 22년 1회 실기 (난이도: ★★★)

다음 Java 코드 중에서 밑줄에 들어갈 알맞는 코드를 작성하시오.

▶ 정답 Car

Car 클래스

```
1  class Car implements Runnable{
2      int a;
3
4      public void run() {
5          system.out.println("message")
6      }
7  }
```

Thread 클래스에서 Runnable 인터페이스를 구현받은 객체를 받아서 run() 메소드를 비동기로 수행할 수 있게 돼요.
여기서는 Car 클래스의 [4행] run() 메소드가 [12행] start() 호출 시, 비동기로 수행돼요.
(비동기는 백그라운드에서 돈다고 생각하면 편해요. 기존 로직 흐름과는 별도로 동작해요.)

실제 실행코드

```
8
9  public class Main{
10     public static void main(String args[]) {
11         Thread t1 = new Thread(new ___());
12         t1.start();
13     }
14 }
```

new Tread의 생성자 안에는 Runnable의 구현체가 들어가야해요.
그러므로, 정답은 **Car**에요!
(물론, 아무것도 없이 Thread를 생성하는 것도 가능은 해요! 지금은 그건 고려하지 않아도 돼요)

[11행] 위에서 별도로 동작시키고 싶은 thread를 생성했고,
start() 메소드를 호출해야 Runnable의 run() 메소드가 새롭게 생긴 thread에서 동작해요.
(여기 코드와는 상관없지만 저 t1.start() 메소드가 종료되길 기다리고 싶으면 t1.join()을 호출하면 돼요! 그럼 Runnable의 run()이 다 수행될 때까지 기다려요!)

★ key point
이 문제는 Thread 클래스를 써보지 않은 사람은 사실 알기 어려운 정보에요.
Thread 라는 클래스의 생성자 매개변수가 어떤 타입인지를 알고 있어야 하기 때문이죠. 문제에서는 Car 클래스 하나만 정의 되어 있으니,
이런 문제가 나온다면 유연하게 Car를 정답으로 쓰는 것도 방법이에요!

★ 어려울 수 있어요.
알고 보면, 쉬운데 컴파일 오류를 찾는 문제를 시험장에서 만나면 어려울 수 있어요.

컴파일 오류를 찾는 문제가 나오면, 먼저 static 메소드에 일반적인 클래스 변수를 사용하고 있는지 확인해주세요. Static 메소드에서는 static 변수만 사용가능해요. static이 없는 클래스 변수를 사용하고 있다면, 그 행이 오류 원인이에요.

▶ 7번 라인

[기출 예제] 23년 3회 실기 (난이도: ★★★)

다음 자바 코드를 실행할 경우 에러가 발생이 된다. 에러가 발생하는 라인명을 작성하시오.

Person 클래스

```
1  class Person {
2      private String name;
3      public Person(String val) {
4          name = val;
5      }
6      public static String get() {
7          return name;
8      }
9      public void print() {
10         System.out.println(name);
11     }
12 }
```

실제 실행코드

```
13 public class main {
14     public static void main(String[] args) {
15         Person obj = new Person("Kim");
16         obj.print();
17     }
18 }
19
```

Person 클래스의 멤버 변수

Person 클래스의 static 메소드

name을 출력해요.

코드가 실행되기 전에 컴파일 오류가 발생합니다.

★ 왜? static메소드에서 멤버 변수는 사용 불가능 할까요?
암기가 아닌, 이해를 하고 싶으신 분들은 보세요!
오른쪽 설명을 먼저 읽어 보시고 읽으면 좋아요

static은 객체를 생성하지 않고도 호출할 수 있어요.
예를 들어, Person.get() 으로 호출하는 게 가능해요.

먼저, Person의 name은 언제 할당될까요?
Person a = Person("kim"); **이렇게 Person 객체를 생성**될 때, name 의 값이 할당 되요.

(1) static 메소드는 객체 생성 없이도 사용할 수 있다 했는데,
(2) 그런 static 메소드 안에서 객체가 생성되어야만 값이 할당되는 멤버 변수(name과 같은)를 사용하려고 하면 당연히 오류가 날 수 밖에요!

위의 이유 말고도, Person a = Person("Kim"); 와 Person b = Person("Lee"); 이렇게 두개의 객체가 선언되고
static 메소드인 Person.get()을 호출하려고 했다고 가정해볼게요.
클래스에 속해 있는 get()은 어떤 객체의 name을 반환해줘야할까요?
(**선택할 수 없죠**)
이러한 이유들로 static 메소드에서는 멤버변수를 사용할 수 없어요

★ key point
static 메소드에서는 **static(정적) 변수**만 사용 가능해요.
(멤버 변수는 사용 불가능)

★static
자바 프로그래밍에서 'static'을 사용하면, 그 요소는 특정 객체에 속하지 않고, 해당 클래스의 모든 객체가 공유합니다.

서로 다른 객체라도 static으로 선언된 변수는 그 클래스의 모든 인스턴스에서 공유되어, 어떤 객체가 그 값을 변경하면 다른 모든 객체에서도 그 변경된 값을 볼 수 있어요.

'static' 메소드도 클래스의 특정 인스턴스가 아니라 클래스 자체에 속해 있기 때문에, 객체를 생성하지 않고도 호출할 수 있습니다. 이런 특성 때문에 'static'은 클래스 수준에서 공유되는 변수나 메소드를 만들 때 유용하게 사용됩니다.
틀린 표현이지만, 이해가 쉽게 표현하자면 C언어의 포인터 처럼 모두 하나의 주소를 바라보는 것과 비슷해요.
Ex)

GV80에서 GV70으로 값이 변경돼요

```
class Car {
    static String name;
    Car(String n) { name = n; }
}
Car c1 = new Car("GV80")
Car c2 = new Car("GV70")
Car.name == c1.name == c2.name == GV70
```

★멤버 변수
클래스 안에 선언된 변수를 의미해요.

이 변수들은 클래스의 객체가 생성될 때마다 **해당 객체**와 연결되어 있습니다.

▶ 정답
S

[기출 예제] 24년 2회 실기 (난이도: ★☆☆)

다음은 Java에 대한 문제이다. 아래 코드를 확인하여 알맞는 출력 값을 작성하시오.

```
1  class Main {
2      public static void main(String[] args) {
3          String str = "ITISTESTSTRING";
4          String[] result = str.split("T");
5          System.out.print(result[3]);
6      }
7  }
```

'str'이라는 변수를 만들고, 그 안에 "ITISTESTSTRING"이라는 텍스트를 저장해요.
'str'에 저장된 문자열을 'T'를 기준으로 나누고, 그 결과를 'result'라는 배열에 저장해요.
'split' 메서드는 문자열을 특정 문자를 기준으로 나누는 기능을 해요.
이 경우, "ITISTESTSTRING"을 'T'로 나누면 다음과 같이 돼요.
["I", "IS", "ES", "S", "RING"]
index 3은 4번째 위치한 S가 출력되게 됩니다.

[기출 예제] 24년 3회 실기 (난이도: ★☆☆)

다음은 Java 코드에 대한 문제이다. 아래 코드를 확인하여 알맞는 출력값을 작성하시오.

▶ 정답
101

실제 실행코드

```
1  public class Main {
2      public static void main(String[] args) {
3          int sum = 0;
4          try {
5              func();
6          } catch (NullPointerException e) {
7              sum = sum + 1;
8          } catch (Exception e) {
9              sum = sum + 10;
10         } finally {
11             sum = sum + 100;
12         }
13         System.out.print(sum);
14     }
15
16     static void func() throws Exception {
17         throw new NullPointerException();
18     }
19 }
```

sum 변수를 선언하고 0을 할당해요.
NullPointerException을 던져요.
NullPointerException이 던져지면, 해당 블록을 실행해요.
func()에서 NullPointerException을 던졌기 때문에 sum + 1을 실행해요. (sum = 1)
만약, NullPointerException에 대한 catch 문이 없었다면 해당 블록에 들어가겠지만,
위에서 NullPointerException에 대한 catch 문이 존재하기 때문에 해당 블록은 무시돼요.
finally는 try의 성공 실패 여부에 상관없이 무조건 거치는 블록이에요.
그렇기 때문에 sum에 100을 추가로 더해줘요.
101이 출력됩니다.

★ key point
예외 처리(Exception Handling)에 대한 이해를 확인하려고 해요.
특히, 예외가 발생했을 때 try, catch, finally 블록이 어떻게 동작하는 지와, 예외 종류에 따라 적절한 처리가 이루어지는지를 알아보는 문제예요.

특정 예외 처리를 올바르게 했는지:
예외의 종류(NullPointerException, Exception)에 따라 적절한 catch 블록이 실행되는지.

finally의 동작:
예외 발생 여부와 상관없이 finally 블록이 항상 실행된다는 것을 아는지.

[기출 예제] 24년 3회 실기 (난이도: ★★★)

다음은 Java 코드에 대한 문제이다. 아래 코드를 확인하여 알맞는 출력값을 작성하시오.

▶ 정답
B0

실제 실행코드

```
1  class Main {
2      public static class Collection<T> {
3          T value;
4          public Collection(T t) {
5              value = t;
6          }
7          public void print() {
8              new Printer().print(value);
9          }
10         class Printer {
11             void print(Integer a) {
12                 System.out.print("A" + a);
13             }
14             void print(Object a) {
15                 System.out.print("B" + a);
16             }
17             void print(Number a) {
18                 System.out.print("C" + a);
19             }
20         }
21     }
22     public static void main(String[] args) {
23         new Collection<>(0).print();
24     }
25 }
```

value라는 이름의 제네릭 타입(T)의 변수를 선언해요.
객체를 만들 때 입력받은 값 t를 value에 저장해요.
새로운 Printer 객체를 만들고, 그 객체의 print 메서드를 호출하면서 value를 넘겨요.
value의 실제 타입(T)에 따라 Printer 클래스의 어떤 print 메서드가 호출될지가 결정돼요.

0이 전달되었으므로 T는 Integer로 추론돼요.
하지만, 중요한 점은 제네릭 타입 소거(type erasure)라는 개념이에요.
컴파일 시점에 제네릭 타입 정보가 소거되면서, T는 실제로 Object로 변환돼요.
그래서 value의 실제 런타임 타입은 Object가 되고,
print 메소드를 호출할 때 Object 버전의 오버로드된 메소드가 선택되는 거예요.
그래서, B0이 출력돼요.

★ key point
메서드 오버로딩(Method Overloading)과 **동적 바인딩(dynamic binding)**의 작동 방식을 이해하고 있는지 확인하려고 해요.
특히, **컴파일 시점과 실행 시점에서의 메서드 호출 결정**이 어떻게 이루어지는지 알아보는 문제예요.
사실상, 틀리기 쉬운 꼬아서 낸 어려운 문제에요. 제너릭에 대한 깊은 이해도를 필요로 하는 문제에요.
이해하지 못하더라도, 비슷하게 나온다면 침착하게 풀면 맞출 수 있어요.

메서드 오버로딩의 동작 방식:
컴파일러는 **메서드의 입력 타입**에 따라 어떤 메서드를 호출할지 결정해요.

제네릭 타입의 한계:
제네릭 타입은 컴파일 시점에 타입 추론만 하고, **실행 시점에는 구체적인 타입 정보를 유지하지 않아요.**

오버로딩된 메서드에서 우선순위:
컴파일러는 입력 타입에 가장 적합한 메서드를 선택하지만, 더 구체적인 타입이 아니라면 Object 메서드가 선택될 수 있다는 점을 이해해야 해요.

[기출 예제] 25년 1회 실기 (난이도: ★☆☆)

▶ 정답
출력1출력5

다음은 Java 코드에 대한 문제이다. 아래 코드를 확인하여 알맞는 출력값을 작성하시오.

```
1  public class Main {
2
3    public static void main(String[] args) {
4
5      int a=5,b=0;
6
7      try{
8        System.out.print(a/b);
9      }catch(ArithmeticException e){
10       System.out.print("출력1");
11     }catch(ArrayIndexOutOfBoundsException e) {
12       System.out.print("출력2");
13     }catch(NumberFormatException e) {
14       System.out.print("출력3");
15     }catch(Exception e){
16       System.out.print("출력4");
17     }finally{
18       System.out.print("출력5");
19     }
20   }
21 }
```

8 → b가 0이므로 "*0으로 나눌 수 없다*"는 **산술 예외(ArithmeticException)**가 발생해요.

9 → 산술 예외가 일어났을 때 이 줄이 실행되어 "출력1"을 화면에 찍어요.

11 → 배열 인덱스를 잘못 사용했을 때 잡는 예외 블록이므로, 해당 블록에는 도착하지 않아요.

13 → 문자열을 숫자로 바꿀 때 형식이 맞지 않으면 잡는 예외 예외 블록이므로, 해당 블록에는 도착하지 않아요.

15 → 위 세 가지 예외 외에 모든 예외를 포괄적으로 잡아내는 마지막 블록이에요. [9행] 예외에 걸렸기에 해당 블록에 도착하지 않아요.

17 → 예외 발생 여부와 상관없이 반드시 실행되는 블록이에요.
"출력5"를 찍습니다.
try가 성공하거나, catch가 실행된 뒤에도 항상 실행돼요.
두 출력을 이어서 출력되어 출력1출력5 가 출력돼요.

★ key point

예외(Exception) 문제는 자주 출제되지 않는 만큼 개념적인 부분이 출제되었습니다.
예외의 흐름과 개념 정도만 알아두시면 좋습니다.

- **0으로 나누기**: ArithmeticException 발생
- **catch 블록**은 위에서부터 순서대로 실행돼요.
- **마지막 catch(Exception e)**로 나머지 예외를 한꺼번에 처리할 수 있어요.
- **finally** 블록은 예외 발생 여부와 관계없이 항상 실행돼요.

[문제 유형] 연산자

▶ 정답
[5행] m / 1000
[6행] (m % 1000) / 500
[7행] (m % 500) / 100
[8행] (m % 100) / 10

[기출 예제] 23년 2회 실기 (난이도: ★★☆)

다음은 JAVA 코드 문제이다.
가지고 있는 돈이 총 4620원일 경우
1000원, 500원, 100원, 10원의 지폐 및 동전을 이용하여 보기의 조건에
맞춰 최소한의 코드를 통해 괄호안을 작성하시오.

[보기]
아래 주어진 항목들을 갖고 괄호안의 코드를 작성
변수 : m
연산자 : / , %
괄호 : [,], (,)
정수 : 1000, 500, 100, 10

실제 실행코드

```
1  public class Problem{
2      public static void main(String[] args){
3          m = 4620;
4
5          a = (              );
6          b = (              );
7          c = (              );
8          d = (              );
9
10         System.out.println(a); // 천원짜리   4장 출력
11         System.out.println(b); // 오백원짜리 1개 출력
12         System.out.println(c); // 백원짜리   1개 출력
13         System.out.println(d); // 십원짜리   2개 출력
14     }
15 }
16
```

4620원을 1000원으로 가질 수 있는 **최대한의 수**를 구해야 해요.
4620원이 1000원으로 나누면 **4** 몫과 **620**의 나머지가 나와요.
몫만 필요하고, 나머지는 필요 없으니,
4620 / 1000으로 표현할 수 있어요.
즉, a에 들어갈 값은 **m / 1000** 이에요

위의 계산으로 1000원짜리로 4장 가지고 있고,
나머지 금액(620원) 중 500원 동전으로 최대 몇 개 가지고 있을 수 있는지 계산하는 식을 구하면 돼요.
(4620원을 1000원으로 나누고 남은 나머지) 를 500원으로 나눈 몫 을 구해야해요.
(m % 1000) / 500 으로 표현할 수 있어요.

위의 계산으로 1000원 4장, 500원 하나를 가지고 있고,
남은 나머지(120원) 중 100원으로 최대 몇 개 가지고 있을 수 있는지 계산하는 식을 구하면 돼요.

최소한의 코드로 작성하라했으니,
(4620원을 500원으로 나누고 남은 나머지) 를 100원으로 나눈 몫으로 표현할 수 있어요.
(m % 500) / 100 으로 표현할 수 있어요.

위의 계산으로 1000원 4장, 500원 하나, 100원 하나를 가지고 있고, 나머지 금액(20원) 중 10원으로 최대 몇 개 가지고 있을 수 있는지 계산하는 식을 구하면 돼요.

최소한의 코드로 작성하라했으니,
(4620원을 100원으로 나누고 남은 나머지) 를 10원으로 나눈 몫으로 표현할 수 있어요.
(m % 100) / 10 으로 표현됩니다.

★ 연산자

/ 연산자 (나눗셈 연산자): 이 연산자는 두 숫자를 나누는 데 사용됩니다. 예를 들어, 10 / 3은 10을 3으로 나눈 값, 즉 3.3333... (소수점 이하의 숫자가 있을 경우) 또는 정수 나눗셈인 경우 3 (소수점 이하를 버림)의 결과를 줍니다.

% 연산자 (나머지 연산자): 이 연산자는 두 숫자를 나눈 후 남는 '나머지'를 구하는 데 사용됩니다. 예를 들어, 10 % 3은 10을 3으로 나눈 후 남는 나머지인 1을 결과로 줍니다.

[기출 예제] 22년 1회 실기 (난이도: ★★☆)

아래 자바 코드에서 출력되는 값을 작성하시오.

▶ 정답 2000

A 클래스

실제 실행코드

```
1  class A {
2      int a;
3      int b;
4  }
5
6  public class Main {
7
8      static void func1(A m){
9          m.a *= 10;
10     }
11
12     static void func2(A m){
13         m.a += m.b;
14     }
15
16     public static void main(String args[]){
17         A m = new A();
18
19         m.a = 100;
20         func1(m);
21         m.b = m.a;
22         func2(m);
23
24         System.out.printf("%d", m.a);
25     }
26 }
```

A 클래스의 a 멤버 변수에 10을 곱해줍니다.
[20행]에서 호출하여, m.a에 m.a * 10을 곱하여 1000이 됩니다.

A 클래스의 a 멤버 변수에 m.b를 더해줍니다.
[22행]에서 호출하여, m.a에 m.a + m.b를 하여 2000이 됩니다.

변수 m에 A 클래스를 선언 및 할당합니다.

m.a에 100을 할당해요.
[9행]을 수행해요. m.a가 1000이 되었어요.
m.b에 1000을 할당해요.
[13행]을 수행해요. m.a가 2000이 되었어요.

%d는 숫자를 치환해줘요.
m.a가 출력되서 정답은 2000이에요.

★ 할당 연산자

- 더하기 할당 연산자

변수의 현재 값에 어떤 값을 더한 후,
그 결과를 다시 변수에 할당해요.
변수 = 변수 + 값 (ex. a = a + 1) 이라면,
변수 += 값 (ex. a += 1)로 표현할 수 있어요.

- 곱하기 할당 연산자

변수의 현재 값에 어떤 값을 곱한 후,
그 결과를 다시 변수에 할당해요.
변수 = 변수 * 값 (ex. a = a * 1) 이라면,
변수 *= 값 (ex. a *= 1)로 표현할 수 있어요.

▶ 정답
1. n > 0, n >= 1, i < 8, i <= 7 (넷중 아무거나 써도 정답이에요)
2. n % 2

[기출 예제] 20년 4회 실기 (난이도: ★★☆)

다음은 n이 10일 때, 10을 2진수로 변환하는 자바 소스 코드이다. 1,2에 알맞는 값을 적으시오.
[출력결과] : 00001010

실제 실행코드

```
1  public class Main {
2      public static void main(String[] args) {
3          int[]a = new int[8];
4          int i=0; int n=10;
5          while (  1.  ) {
6              a[i++] = (  2.  );
7              n /= 2;
8          }
9
10         for(i=7; i>=0; i--) {
11             System.out.print(a[i]);
12         }
13     }
14 }
```

8 크기의 빈 배열을 생성했어요.

★ i++ (후위 증가 연산자)
현재 i의 값을 먼저 사용한 후에, i의 값을 1 증가시켜요.
예를 들어, int j = i++;라고 하면, 먼저 j에 i의 현재 값이 할당되고, 그 다음에 i의 값이 1 증가해요.

따라서 i가 초기에 5였다면,
j에는 5가 할당되고, i는 6이 되는 거예요.

코드의 흐름대로 천천히 따라가면 어렵지 않아요.
[6행]에서 i가 0부터 +1씩 증가하면서 값을 채우고 있는데,
[5행]에서 while문의 조건식을 어떻게 해야 [3행]에서 선언한 8번만 반복할 수 있을까요?

여러가지 방법이 있지만, 간단하게 **1번의 정답은 i < 8 이나 i <= 7** 을 적을 수 있어요.
i가 8 미만 (7보다 같거나 작음) 까지만 동작하도록 1번 정답을 적으면 돼요.

그 다음으로는 먼저, [10행]을 살펴봐요.
[10행]의 출력이 거꾸로 찍히고 있어요.
[4행]의 i가 0부터 배열을 채워넣고 있으니
우리는 거꾸로해서 01010000의 순서로 [6행] 2번 정답 코드에서 할당되어야 해요.

[7행]의 n /= 2는 n = n / 2라고 표현할 수 있어요.
이는 n을 계속 2로 나눈 몫을 덮어쓰는 거에요.
여기서는 10 -> 5 -> 2 -> 1 -> 0 -> 0 -> … 이렇게 진행 될거에요.

근데, n의 값을 살펴보니 2번째와 4번째만 홀수고 나머지는 전부 짝수네요?
만약에 짝수는 0 홀수는 1로 값을 a에 할당 할 수만 있다면
01010000으로 배열이 채워 질거에요.

짝수는 0 홀수는 1로 표현되는 **2번의 정답은 n % 2**이 들어가야해요.
%는 나누고 남은 나머지를 반환하거든요
예를 들어,
10 % 2의 나머지는 0
5 % 2의 나머지는 1
2 % 2의 나머지는 0
이런식으로요.

★ /= 연산자
나눗셈 후 대입 연산자예요.
변수의 값을 어떤 수로 나눈 후,
그 결과를 다시 같은 변수에 대입해요.

예를 들어, a /= 3은 a의 값을 3으로 나눈 후, 그 결과를 다시 a에 저장한다는 뜻이에요.
만약 a가 10이었다면, a /= 3 실행 후에는 a의 값이 3이 돼요. (10 / 3의 몫은 3)

★ %= 연산자
%=는 나머지 연산 후 대입 연산자예요.
변수의 값을 어떤 수로 나눈 후,
그 나머지를 다시 같은 변수에 대입해요.

예를 들어, a %= 3은 a를 3으로 나눈 나머지를 구한 후, 그 결과를 다시 a에 저장한다는 뜻이에요.
만약 a가 10이었다면, a %= 3 실행 후에는 a의 값이 1이 돼요. (10을 3으로 나눈 나머지는 1)

★while 루프
특정 조건이 참인 동안 코드 블록을 반복 실행하는 루프예요.
조건이 참인 동안에는 코드 블록 안의 내용이 계속해서 실행돼요.
조건이 거짓이 되면 루프를 빠져나와요.

예를 들어, while (count < 5)는 count가 5보다 작은 동안에는 루프 안의 코드를 계속 실행한다는 뜻이에요.

[기출 예제] 21년 3회 실기 (난이도: ★★☆)

다음 Java 코드에 대한 알맞는 출력값을 쓰시오.

▶ 정답 7

```
1  public class testco {
2      public static void main(String[] args) {
3          int a = 3, b = 4, c = 3, d = 5;
4          if ((a == 2 | a == c) & !(c > d) & (1 == b ^ c != d)) {
5              a = b + c;
6              if (7 == b ^ c != a) {
7                  System.out.println(a);
8              } else {
9                  System.out.println(b);
10             }
11         } else {
12             a = c + d;
13             if (7 == c ^ d != a) {
14                 System.out.println(a);
15             } else {
16                 System.out.println(d);
17             }
18         }
19     }
20 }
```

조건문

그 외

변수 a, b, c, d에 각각 3, 4, 3, 5라는 초기값을 할당했어요.

이 if 문은 여러 개의 조건을 비트 연산자로 연결하고 있어요.
어려우니 하나씩 하나씩 넘어가 볼게요.

(a == 2 | a == c)
a가 2이거나 a가 c와 같은지 확인해요.
여기서는 | (OR) 연산자를 사용했어요.
a와 c가 같으니 참이에요.

만약 처음 if 문의 조건을 만족하지 않으면 else 문으로 들어가요.
a에 c와 d를 더한 값을 할당해요.

!(c > d)
c가 d보다 크지 않은지 확인해요.
!(NOT) 연산자를 사용했어요.
c가 d보다 크지 않아 참이에요.

또 if 문이 나와요.
이번에는 7 == c ^ d != a라는 조건을 확인해요.
7 == c 와 d != a 의 결과값이 달라야 참
같으면 거짓이에요.
여기서는 해당 조건문을 타지 않아요.

(1 == b ^ c != d)
1이 b와 같은지, 그리고 c가 d와 다른지 확인해요.
여기서는 ^(XOR) 연산자를 사용했어요.
1==b의 결과와 c!=d의 결과가 달라야 참이에요.
false ^ true로 두 결과가 다르니 참이에요.

모든 조건을 만족해 안으로 들어가요.

a에 b와 c를 더한 값을 할당해요. a는 7이 되었어요.

그리고 또 다른 if 문이 나와요.
이번에는 7 == b ^ c != a라는 조건을 확인해요.
만약 이 조건이 참이면 a 값을 출력하고, 거짓이면 b 값을 출력해요.
7 == b(5) 이므로 거짓, c(3) != a(7) 이므로 참이에요.
둘의 결과가 달라 참을 반환해요.

[4행] [6행]이 참이므로 **정답은 7**이 돼요.

★ 비트 연산자
이 코드에서 비트 연산자가 조금 어려울 수 있어요.

| (OR)는 두 조건 중 하나라도 참이면 참이 되는 연산자예요.
^ (XOR)는 두 조건이 서로 다를 때만 참이 되는 연산자예요.
! (NOT)은 조건의 반대를 나타내는 연산자예요.

이해하기 어려울 수 있지만, 비트 연산자를 사용하면 여러 조건을 효율적으로 확인할 수 있어요.

[기출 예제] 20년 4회 실기 (난이도: ★★☆)

다음은 자바 소스 코드이다. 출력 결과를 보고 , 1,2에 알맞는 값을 적으시오.

▶ 정답
1) 3
2) 5

[출력 결과]
1 4 7 10 13
2 5 8 11 14
3 6 9 12 15

실제 실행코드

```
1  public class Main {
2      public static void main(String[] args) {
3          int[][]a = new int[(1.)][(2.)];
4          for(int i = 0; i <3; i++) {
5              for(int j=0; j < 5; j++) {
6                  a[i][j] = j*3+(i+1);
7                  System.out.print(a[i][j]+"");
8              }
9              System.out.println();
10         }
11     }
12 }
```

사실 코드를 보면 어려워 보이지만, 바로 정답을 찾을 수 있는 요소들이 있어요.
[6행]을 계산하려고 보면 어렵겠지만, [6행]을 볼 필요가 없어요.

아래의 생각의 흐름을 따라 정답을 유추해볼게요.
출력 결과를 보면 2차원 배열처럼 보이네요.
행이 3개 열이 5개 인 것 만으로 사실 **1번 정답은 3, 2번 정답은 5**가 바로 나와요.

근데, 3,5 인지 5,3인지 헷갈릴 수 있어요.
그럴 때는 **for문**을 살펴보면 돼요.

[4행] for문의 i는 3보다 작을 때까지 반복해요.
근데, 그 i가 a[i][j] 왼쪽에 들어가 있는 걸 확인 할 수 있어요.
[5행] for문의 j는 5보다 작을 때까지 반복하는데 두 번째 배열에 j가 들어가 있죠.

이런 두가지 조건들만 가지고도 빠르게 정답을 유추하여 정답을 적을 수 있어요.

[기출 예제] 23년 1회 실기 (난이도: ★☆☆)

아래 자바 코드에서 출력되는 값을 작성하시오.

▶ 정답
[17행] 10
[18행] 11
[19행] 10
[20행] 20

Static 클래스

```
1  class Static{
2
3      public int a = 20;
4      static int b = 0;
5
6  }
```

- Class 이름이 Static 인 것! 따로 기능이 있는 것이 아니에요
- Class 의 멤버 변수에요. 객체별로 따로 가지는 변수에요
- Class 의 static 변수에요. 모든 인스턴스가 공유하는 변수에요.

실제 실행코드

```
8  public class Main{
9      public static void main(String[] args) {
10
11         int a;
12         a = 10;
13         Static.b = a;
14
15         Static st = new Static();
16
17         System.out.println(Static.b++);
18         System.out.println(st.b);
19         System.out.println(a);
20         System.out.print(st.a);
21     }
22 }
```

- a 라는 지역 변수를 생성했어요. 지금은 값이 없어요.
- a 에 10을 할당했어요.
- Static Class의 static 변수 b에 a(10)을 할당했어요.
- 모든 Static 클래스의 인스턴스들은 같은 b 값을 공유해요.
- st 변수에 Static 객체를 생성 후 할당했어요.
  지금은, st 의 변수 b도 동일하게 10을 가지고 있어요.
- 모두가 공유하는 Static 클래스의 b 변수를 반환한 다음 +1 을 했어요.
  Static.b는 11이 되었어요.
- st.b는 Static.b와 동일한 변수를 바라보고 있어요. 위에서 10에 +1을 했기에 11을 반환해요.
- a의 값은 [12행]에서 할당한 10이에요
- st.a의 값은 [15행]에서 생성된 객체의 a이므로, 20이에요. 멤버변수 a를 반환하고 있어요.
  멤버변수는 객체마다 각자의 변수를 가져요.

★ ++ 연산자

전위 증가 (예: ++변수)
이 방식에서는 먼저 변수의 값이 1 증가하고, 그 후에 이 값이 사용됩니다.

예를 들어, int a = 5; int b = ++a;라고 하면, a는 6이 되고, b도 6이 됩니다.

후위 증가 (예: 변수++)
이 방식에서는 변수의 현재 값이 먼저 사용되고, 그 후에 변수의 값이 1 증가합니다.

예를 들어, int a = 5; int b = a++;라고 하면, b에는 a의 원래 값인 5가 저장되고, 그 후에 a는 6이 됩니다.

★ 지역변수

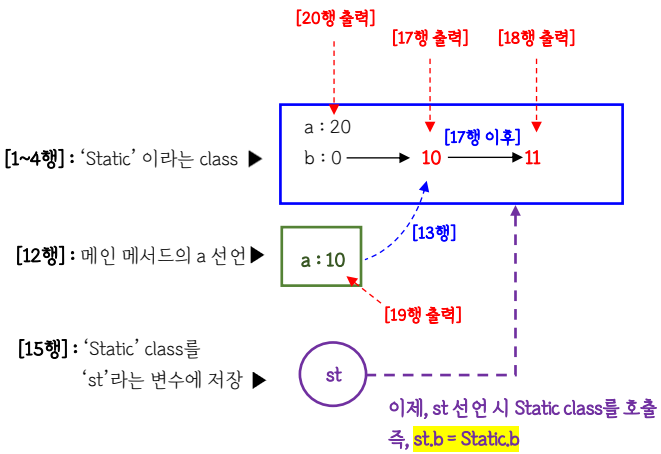
지역변수는 현재 진행되는 메소드 안에서만 생성된 변수를 말해요.

지역변수는 메소드 안에서만 살아있고, 메소드가 종료되면 사라져요.

지역변수는 해당 메소드 외부에서는 접근(읽고, 쓰기) 할 수 없어요.

동일한 이름의 변수가 멤버 변수도 선언되어 있을 때, 지역 변수를 사용해요.
(이때, 멤버 변수는 shadowed 되었다고 해요.)

★ 아래 도식표에서 행 순서대로 따라가보세요!



[문제 유형] 재귀 함수

[기출 예제] 24년 2회 실기 (난이도: ★★☆)

다음은 Java에 대한 문제이다. 아래 코드를 확인하여 알맞는 출력 값을 작성하시오.

▶ 정답
dcba

잠깐! calculFn 의 [11행]을 보고 재귀 함수인 것을 먼저 캐치해야 해요!

```
1  class Main {
2      public static void main(String[] args) {
3          String str = "abacabcd"; ⟶ "abacabcd"라는 문자열을 'str'이라는 변수에 저장합니다.
4          boolean[] seen = new boolean[256];⟶256개의 false 값으로 초기화된 배열을 만들어요. 각 문자가 이미 처리되었는지 여부를 추적하는 데 사용해요.
5          System.out.print(calculFn(str, str.length()-1, seen));⟶'calculFn' 함수를 호출하고 그 결과를 화면에 출력해요.
6      }                                                          이 함수에 문자열, 문자열의 마지막 인덱스, 그리고 'seen' 배열을 전달해요.
7
8      public static String calculFn(String str, int index, boolean[] seen) {
9          if(index < 0) return ""; ⟶인덱스가 0보다 작으면 빈 문자열을 반환해요.
10         char c = str.charAt(index);⟶인덱스의 문자를 가져와요.
11         String result = calculFn(str, index-1, seen);⟶인덱스를 하나 줄여서 다시 호출해요. (재귀 호출)
12         if(!seen[c]) {⟶현재 문자가 아직 처리되지 않았다면,
13             seen[c] = true;⟶ 처리됨 것으로 표시
14             return c + result;⟶현재 문자를 결과 문자열의 앞에 추가해요.
15         }
16         return result;⟶현재 문자가 이미 처리되었다면, 결과를 그대로 반환해요.
17     }                          즉, 이 메소드는 주어진 문자열에서 중복된 문자를 제거하고, 마지막으로 등장한 순서대로 문자들을 나열해요.
18 }                              abacabcd 문자열을 뒤에서 부터 중복 제거하면 dcba가 출력돼요
```

★ 재귀 함수

재귀 함수란 자기 자신을 호출하는 함수예요.
종료 조건을 만나기 전까지 계속 자기 자신을 호출하면서 종료 조건을 만나면 값을 반환해요.
큰 문제를 같은 형태의 작은 문제로 나누어 해결하는 방식이에요.

[기출 예제] 25년 1회 실기 (난이도: ★★☆)

다음은 Java에 대한 문제이다. 아래 코드를 확인하여 알맞는 출력 값을 작성하시오.

▶ 정답 20

```
1  public class Main {
2
3      public static void main(String[] args) {
4          int[] nums = {3, 5, 8, 12, 17};
5          System.out.println(maxSum(nums, 0, nums.length - 1));
6      }                                      5(배열 크기) – 1 = 4를 넣어 주고 있어요.
7
8      static int maxSum(int[] nums, int start, int end) {
9          if (start >= end) return 0; ⟶ 종료 조건 (더 이상 쪼갤 수 없음)
10         int mid = (start + end) / 2;⟶시작과 끝의 중간을 구함
11         return nums[mid] + Math.max(maxSum(nums, start, mid), maxSum(nums, mid + 1, end));
12     }  [11행]에서 이 함수는 중간 값 + max(왼쪽 배열, 오른쪽 배열) 값을 구하고자 함을 우리는 알았어요.
13 }       이렇게 배열이 작은 경우 값을 대입해보는 게 쉽게 문제 풀 수 있어요.
```

함수의 의미 자체는 배열을 반으로 나눌 때마다 **그 구간의 중간(피벗) 값을 더해** 가면서,
왼쪽과 오른쪽 중 '**더 큰 쪽**'을 선택해 내려갔을 때 얻을 수 있는 **합의 최댓값**을 구하자하는 문제에요.
(대입해서 푸는 게 맞기 때문에 함수의 목적을 알 필요는 없어요)

n[0]	n[1]	n[2]	n[3]	n[4]
3	5	8	12	17

종료 조건인 start와 end의 값이 같을 때까지 식을 정리해보면 돼요.
maxSum(0,4) = n[2] + max(maxSum(0,2), maxSum(3,4))
maxSum(0,2) = n[1] + max(maxSum(0,1), maxSum(2,2))
maxSum(0,1) = n[0] + max(maxSum(0,0), maxSum(1,1))

좌측 식이 모두 정리되었어요. start, end가 같은 걸 0으로 치환해서 계산해보면 다음과 같아요.
maxSum(0,1) = 3 + max(0,0)
maxSum(0,2) = 5 + max(3,0)
maxSum(0,4) = 8 + max(5, maxSum(3,4))

이제 maxSum(3,4)만 구해서 계산하면 돼요.
maxSum(3,4) = n[3] + max(maxSum(3,3), maxSum(4,4))
maxSum(3,4) = 12 + max(0,0) = 12

즉, maxSum(0,4) = 8 + max(5,12) = 8 + 12 = 20 이에요.

★ key point
오버로딩과 재귀에 관련된 문제입니다.
오버로딩은 메소드의 매개변수의 타입이나 개수가 다르면 서로 다른 메소드로 취급합니다.

▶ 정답
4

[기출 예제] 25년 1회 실기 (난이도: ★★☆)

다음은 Java 코드에 대한 문제이다. 아래 코드를 확인하여 알맞는 출력값을 작성하시오.

```
1  public class Main {
2    public static void main(String[] args) {
3      System.out.println(calc("5"));
4    }
5
6    static int calc(int value) {
7      if (value <= 1) return value;
8      return calc(value - 1) + calc(value - 2);
9    }
10
11   static int calc(String str) {
12     int value = Integer.valueOf(str);
13     if (value <= 1) return value;
14     return calc(value - 1) + calc(value - 3);
15   }
16 }
```

7 → 1 이하가 되면 그 값을 반환합니다. 재귀 함수의 종료 조건입니다.
12 → 문자열을 정수로 변환합니다. 문자열 5가 숫자 5가 됩니다.
13 → 1 이하가 되면 그 값을 반환합니다. 재귀 함수의 종료 조건입니다.
14 → 이제 정수 매개변수인 calc이 호출됩니다.

calc("5") = calc(4) + calc(2)
calc(4) = calc(3) + calc(2)
calc(3) = calc(2) + calc(1)
calc(2) = calc(1) + calc(0)
calc(1) = 1
calc(0) = 0
calc(2) = 1 + 0 = 1
calc(3) = 1 + 1 = 2
calc(4) = 2 + 1 = 3
calc("5") = 3 + 1 = 4

먼저, 하나씩 분해하고
분해된 값들을 종료조건에 도달하여 값이 나오면
그걸 역순으로 계산하면 정답이 나와요!

정답은 4가 출력돼요!

[문제 유형] 함수 호출

[기출 예제] 20년 3회 실기 (난이도: ★☆☆)

다음 C언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오.
(단, 출력문의 출력 서식을 준수하시오.)

▶ 정답 234

★ 가장 기본이 되는 함수 호출 문제에요.
20년도에 딱 한번만 출제되고, 더 이상 출제되지 않고 있는 문제에요.

```
1  #include <stdio.h>
2
3  int r1() {
4      return 4;
5  }
6
7  int r10() {
8      return (30+r1());
9  }
10
11 int r100() {
12     return (200+r10());
13 }
14
15 int main() {
16     printf("%d\n", r100());
17     return 0;
18 }
```

30 + 4 = 34

234

[16행]을 호출하면, 200 + [7행] -> [8행] 30 + [3행] -> [4행] 4 반환
호출된 순서대로 반환하다보면 4 + 30 + 200 으로 234가 반환되어
정답은 234 에요.

[문제 유형] 완전수

★ 완전수
완전수는 자기 자신을 제외한 모든 양의 약수를 더하면 자기 자신이 되는 수를 말해요.
예를 들어, 6은 완전수인데, 1, 2, 3은 6의 약수이고 이들의 합 (1 + 2 + 3)은 6과 같거든요.
완전수는 100이하에서 **6**과 **28**만 존재해요! 암기하면 정말 좋아요.

[기출 예제] 23년 3회 실기 (난이도: ★☆☆)

▶ 정답 34

다음 C 프로그램 출력 결과 구하기

```c
#include <stdio.h>

int test(int n) {
    int i, sum = 0;

    for (i = 1; i <= n / 2; i++){
        if (n % i == 0)
        sum += i;
    }

    if (n == sum)
        return 1;
    return 0;
}


int main(){
    int i, sum=0;

    for (i = 2; i <= 100; i++){
        if (test(i))
        sum += i;
    }

    printf("%d ", sum);
    return 0;
}
```

정수 n을 받아서, n이 완전수인지 아닌지를 판별해요.
완전수는 자기 자신을 제외한 약수의 합이 자기 자신과 같은 수를 말해요.

for 루프를 사용해서 1부터 n/2까지의 수 i에 대해 반복해요.
만약 n을 i로 나눈 나머지가 0이면 (즉, i가 n의 약수이면), sum에 i를 더해요.

루프가 끝나고 나서, n과 sum이 같으면 1(true)을 반환하고,
그렇지 않으면 0 (false)을 반환해요.

for 루프를 사용해서 2부터 100까지의 수 i에 대해 반복해요.
각 i에 대해 test(i)를 호출해서, i가 완전수인지 확인해요.
만약 test(i)가 참 (즉, 1)을 반환하면, sum에 i를 더해요.
루프가 끝나고 나서, printf 함수를 사용해서 sum을 출력해요.

2~100사이의 완전수는
6(1+2+3)과 28(1+2+4+7+14) 두개에요.
그래서 정답은 6과 28의 합인 34에요.

★ for 구문
for 문은 일종의 반복적인 작업을 수행하기 위한 지시문이에요. for 문은 크게 **세 부분**으로 나누어 생각할 수 있습니다. 초기화(선언 및 할당), 조건, 그리고 반복 후 작업입니다. 예를 들어, 우리가 1부터 10까지 숫자를 차례대로 출력하고 싶다면, for 문을 사용하여 이 작업을 할 수 있습니다. for 문은 다음과 같은 형식이에요.

```
for (초기화; 조건; 반복 후 작업) {
   // 반복해서 실행할 명령들
}
```

예를 들어, 1부터 10까지 출력하는 for 문은 다음과 같이 작성돼요.

```
for (int i = 1; i <= 10; i++) {
   printf("%d₩n", i);
}
```

이 코드는 i가 1부터 시작해서 10이 될 때까지 i의 값을 출력하며, 각 반복마다 i를 1씩 증가시켜요.

★ [6행] for 문 n=6 일 때
이 부분에서 i는 1에서 시작하여 n / 2 (6 / 2 = 3) 이하일 때까지 반복합니다. 즉, i는 1, 2, 3의 값을 갖게 돼요.
왜 n / 2까지인가? 숫자의 약수는 그 숫자의 절반을 넘을 수 없기 때문입니다. 예를 들어, 6의 경우 3보다 큰 수는 6의 약수가 될 수 없습니다.

반복문 내부에 if (n % I == 0) 조건문이 있습니다. 이는 n을 i로 나눈 나머지가 0인지 확인하여 i가 n의 약수인지를 판단해요.
N이 6일 때, i가 1, 2, 3일 경우에 이 조건은 참이 돼요. 즉, 이들은 모두 6의 약수입니다.
참일 경우, sum += I;를 통해 약수 i의 값을 sum에 더해요. 이렇게 해서 1, 2, 3을 sum에 차례로 더해요.

For 문이 종료된 후, if (n == sum) 조건문을 통해 n이 sum과 같은지 확인합니다. 이 경우, sum은 1 + 2 + 3 = 6이 되므로, n이 6이라면 이 조건은 참이 돼요.
조건이 참이면, 함수는 return 1;을 실행하여 1을 반환해요. 이는 n이 완전수임을 의미해요.

★ key point
손으로 하나 하나 계산하는 건 비효율적이에요.
코드에서 원하는게 완전수의 합인 걸 캐치했다면, 완전수를 찾는 최적화된 과정을 진행해야해요.
(사실, 제일 쉬운 방법은 완전수를 외우는 거에요. 문제에서는 6, 28 그리고 다음 완전수는 496 이거든요. 완전수 문제에서는 100 이하로만 나올 확률이 높아요)

완전수는 짝수예요.
왜냐하면 1을 제외한 모든 완전수의 약수는 짝수-홀수 쌍으로 이루어져 있거든요.
따라서 2부터 시작해서 짝수만 체크하면 돼요.

어떤 수의 약수를 찾을 때, 그 수의 제곱근까지만 확인하면 돼요.
왜냐하면 어떤 수의 약수들은 그 수의 제곱근을 기준으로 대칭을 이루기 때문이에요.
예를 들어, 36의 약수는 1, 2, 3, 4, 6, 9, 12, 18, 36이에요. 36의 제곱근은 6이에요.
1, 2, 3은 6보다 작고, 36, 18, 12는 6보다 커요. 6은 6과 같고요.
따라서 6보다 작거나 같은 약수들만 확인하고, 나머지 약수들은 36을 그 약수들로 나누면 구할 수 있어요.

[기출 예제] 22년 3회 실기 (난이도: ★☆☆)

▶ 정답 2

다음 C언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오.

```c
#include <stdio.h>

void main() {
    int n;
    int k;
    int s;
    int el = 0;

    for(n=6; n<=30; n++){
        s=0;
        k=n/2;
        for(int j=1; j<=k; j++){
            if(n%j==0){
                s=s+j;
            }
        }

        if(s==n){
            el++;
        }
    }

    printf("%d", el);
}
```

이 문제는 6부터 30까지의 정수 중에서 완전수(perfect number)의 개수를 찾는 코드에요.
(우리는 완전수가 6, 28인걸 알고있으니 정답은 2인걸 바로 알 수 있어요)

앞에서 이야기 했듯, 완전수는 100 이하 수에는 6, 28 밖에 없고
굳이 더 적어보자면 6, 28, 496, 8128, 33550336 … 이렇게 있어요.

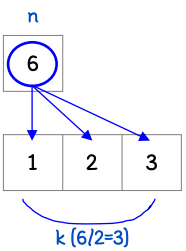
완전수 문제가 나왔을 때, 6과 28 정도는 외워두면 정말 크게 시간을 단축할 수 있으니 외워두는 걸 추천해요!!

그럼, 코드로 이게 **완전수 문제**인지는 어떻게 알 수 있을까요?

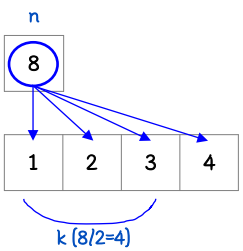
[13행] [14행] [18행]이 중요해요.
[13행]은 n 을 나눴을 때 나머지가 0인 값들에 대해서, 즉 **n의 약수에 대해서**
[14행]은 s에 전부 더하고,
[18행]은 **그 값이 n과 같을 때** 를 의미해요.

위 조건들만 체크하더라도, 손으로 계산을 하나 하나 하지 않아도 정답을 알 수 있어요.
여기서는 6~30 사이의 완전수의 개수를 묻는 문제에요.
6과 28인 2가 출력돼요.

★ 완전수를 손으로 계산하고 싶으면 다음과 같이 계산해볼 수 있어요.



6 % 1 = 0 (6을 1로 나눈 나머지는 0)
6 % 2 = 0 (6을 2로 나눈 나머지는 0)
6 % 3= 0 (6을 3로 나눈 나머지는 0)
1 + 2 + 3 = 6 이므로 완전수



8 % 1 = 0 (8을 1로 나눈 나머지는 0)
8 % 2 = 0 (8을 2로 나눈 나머지는 0)
8 % 3= 2 (8을 3로 나눈 나머지는 2)
8 % 4= 0 (8을 4로 나눈 나머지는 0)
1 + 2 + 4 = 7 이므로 8과 같지 않아요.
완전수가 아닙니다.

[문제 유형] 단순 for문

[기출 예제] 23년 2회 실기 (난이도: ★☆☆)

▶ 정답 BCD

다음 C 프로그램 출력 결과 구하기

```
1  #include <stdio.h>
2
3  void main() {
4      int n[3] = {73, 95, 82};
5      int i, sum = 0;
6
7      for (i = 0; i < 3; i++) {
8          sum += n[i];
9      }
10
11     switch (sum / 30) {
12         case 10:
13         case 9:
14             printf("A");
15         case 8:
16             printf("B");
17         case 7:
18         case 6:
19             printf("C");
20         default:
21             printf("D");
22     }
23 }
```

배열 선언!

n[0]	n[1]	n[2]
73	95	82

i가 3보다 작으면 아래 Sum += n[i] 실행
이후 i++ 실행 (i를 1씩 증가)

i	0	1	2	3
sum	sum+n[0] = 0 + 73 = **73**	73 + n[1] = 73 + 95 = **168**	168 + n[2] = 168 + 82 = **250**	for문 종료

Sum/30 = 250을 30으로 나눴을 때 몫 = **8**
'Case 8'에 해당되는 코드 실행 = **"B" 출력**
이후 Break 없기 때문에 마지막 default까지 실행
최종 'BCD' 출력 (정답)

★ Switch case 구문

```
switch(조건식){
  case 변수1: 실행 코드 1
    break;
  case 변수2: 실행 코드 2
    break;
  …
  case 변수n: 실행 코드 n
    break;
  default:
}
```

조건식 값과 case 변수값이 일치 시 해당 case 코드 실행
(ex. 조건식 = 변수2 → 실행코드 2 실행)

break : 해당 Switch 구문 종료!
break 없을 시 아래 코드로 넘어가 계속 실행

default : 모든 case 변수에 해당 조건이 없을 시 실행하는 코드

[기출 예제] 21년 2회 실기 (난이도: ★☆☆)

▶ 정답 1024

다음 C언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오. 주어진 밑(base)과 지수(exp)를 사용하여 거듭제곱을 계산하는 코드에요
(단, 출력문의 출력 서식을 준수하시오.)

```
1  #include <stdio.h>
2
3  int mp(int base, int exp) {
4      int res = 1;
5      for(int i=0; i < exp; i++) {
6          res *= base;
7      }
8
9      return res;
10 }
11
12 int main() {
13     int res;
14     res = mp(2,10);
15     printf("%d",res);
16     return 0;
17 }
```

두 개의 정수 인자, base와 exp를 받아요. 여기서 base는 밑이고, exp는 지수예요.
res라는 변수를 1로 할당하고,
for 반복문을 사용해 base를 exp번 만큼 곱해요. res = res * base와 같은 표현이에요.
거듭제곱 값인 res를 반환해요.
2의 10제곱을 계산해요.
2를 10번 곱하는 거예요.
즉, 2 x 2 x 2 x 2 x 2 x 2 x 2 x 2 x 2 x 2 = 1024를 계산하는 거죠.
정답은 1024가 돼요.

[기출 예제] 23년 2회 실기 (난이도: ★☆☆)

▶ 정답 505

다음 C언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오.

```
1  #include <stdio.h>
2
3  int main(){
4      int c = 0;
5
6      for(int i = 1; i <=2023; i++) {
7          if(i%4 == 0) c++;
8      }
9      printf("%d", c);
10 }
```

'c'라는 이름의 변수를 만들고, 0으로 할당하고 있어요.

'i'라는 변수를 1부터 시작해서 2023까지 1씩 증가시키면서 반복하고 있어요.
모듈러 연산자를 이용해, 'i'를 4로 나눈 나머지를 구해요.
만약 그 나머지가 0이면, 'c'의 값을 1 증가시키는 거예요.
즉, 'i'가 4의 배수일 때마다 'c'의 값을 1씩 증가시키는 것이죠.
즉, 코드의 핵심 목적은 1부터 2023까지의 수 중에서 4의 배수가 몇 개인지 세는 것입니다.

1~2023 중, 4의 배수는 2023를 4로 나누면 나와요.
2023/4 = 505 이 나와요.
정답은 505를 출력해요.

★ 모듈로(modulo) 연산자

% 연산자는 모듈로(modulo) 연산자라고 불려요.

이 연산자는 두 수를 나눈 후 나머지를 반환해요.
예를 들어, 7 % 3은 1이에요. 7을 3으로 나누면
몫은 2이고 나머지는 1이거든요.

수학적으로 표현하면, a % b = a - (a / b) * b 와 같아요.
여기서 /는 몫을 나타내요.

% 연산자는 주로 다음과 같은 상황에서 사용돼요

어떤 수가 특정 수의 배수인지 확인할 때
예를 들어, x % 2 == 0이면 x는 2의 배수예요. (x는 짝수)
x % 3 == 0이면 x는 3의 배수예요.

어떤 수를 특정 범위 내로 제한할 때
예를 들어, x % 10은 항상 0에서 9 사이의 값이에요.
이런 특성을 이용해서 큰 수를 일정 범위 내의 수로 "축소"할 수 있어요.

순환적인 패턴을 만들 때
예를 들어, 0부터 6까지의 값을 반복하고 싶다면,
x % 7을 사용할 수 있어요.
x가 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10일 때,
x % 7은 0, 1, 2, 3, 4, 5, 6, 0, 1, 2, 3이 돼요.

[기출 예제] 22년 1회 실기 (난이도: ★★☆)

▶ 정답 29

다음 C언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오.
(단, 출력문의 출력 서식을 준수하시오.)

```
1  #include <stdio.h>
2
3  int isPrime(int number) {
4      int i;
5      for (i=2; i<number; i++) {
6          if (number % i == 0) return 0;
7      }
8      return 1;
9  }
10
11 int main(void) {
12     int number = 13195, max_div=0, i;
13     for (i=2; i<number; i++)
14         if (isPrime(i) == 1 && number % i == 0) max_div = i;
15     printf("%d", max_div);
16     return 0;
17 }
```

특정 숫자의 가장 큰 소수인 약수를 찾는 예제예요.
여기서 사용된 개념은 '소수'와 '약수'인데, 개념은 다음과 같아요.

★ 소수(Prime Number)
1과 자기 자신 외에는 어떤 수로도 나눌 수 없는 숫자예요.
예를 들어, 2, 3, 5, 7 등이 소수예요.

★ 약수(Divisor)
한 수를 나누어 떨어지게 하는 숫자를 말해요.
예를 들어 6의 약수는 1, 2, 3, 6이에요.

3행: 주어진 숫자가 소수인지 아닌지 판별해요.
5행: 2부터 (주어진 숫자 - 1)까지
6행: 각 숫자로 주어진 숫자를 나눠봐요.
만약 나누어 떨어지면 (즉, % 연산 결과가 0이면 = 약수가 존재) 그 숫자는 소수가 아니라고 판단하고
0을 반환해요. 그렇지 않으면 1을 반환해요, 즉 그 숫자는 소수예요.
11행: 검사할 숫자예요, 여기서는 13195예요.
12행: max_div는 가장 큰 소수 약수를 저장할 변수예요, 0으로 할당되어 있어요.
13행: 2부터 number까지 반복하면서
14행: 각 숫자가 소수인지(isPrime(i) == 1) 이면서, number의 약수인지(number % i == 0) 확인해요.
15행: 그 숫자가 소수이면서 number의 약수라면, max_div에 그 숫자를 저장하고, 출력해요.

★ 이런 식으로 답을 유추할 수 있어요.
단순 for문 문제지만, 답을 찾는 과정을 유추하기 까다로운 문제에요.
13195의 약수 중 가장 큰 소수를 찾아야 하는 걸 알았다고 하더라도, 이걸 손으로 풀기에는 까다로울 수 있어요.
이때, 조금은 쉽게 찾을 수 있는 방법을 알려드릴게요.

방법은 소수를 이용해서 나눠지지 않을 때까지 나누는 거에요.
13195 / 소수1 => (나온 값) / 소수2 => (나온 값) / 소수3 를 진행하다보면, 더 이상 나눠지지 않는 시점이 오는데
여태 나누는데 사용했던 모든 소수(소수1, 소수2, 소수3…) 중 가장 큰 값이 가장 큰 소수이면서 약수인 값이에요.
13195 / [소수] = 나온 값은 일단, 13195의 약수이면서 그 값이 더 이상 나뉘지 않는다면 가장 큰 소수일 테니까요.

소수는 2, 3, 5, 7, 11 … 이렇게 있어요.
2로도 나뉘어지지 않아요. 넘어가요. (ex. 13195 % 2 = 1의 나머지가 있으니 나눠지지 않아요.)
3로도 나뉘어지지 않아요. 넘어가요. (ex. 13195 % 3 = 1의 나머지가 있으니 나눠지지 않아요.)
5으로 정확하게 나눠지는데, 2,639가 나와요. (지금까지 13195의 약수면서 가장 큰 소수는 5에요)
다음으로는 7로 나뉘지는데, 377가 나와요. (지금까지 13195의 약수면서 가장 큰 소수는 7에요)
그 다음 11로는 나눠지지 않아요. 넘어가요.
13으로 나뉘지는데, 29가 나와요. (13, 29 모두 소수이기 때문에 정답은 29에요)
29는 더 이상 나뉘어 지지 않는 소수이기 때문에 29가 정답이에요.

★ 소수

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

★ Key Point
1. 문자의 ASCII 코드 값을 이용한 값 차이
2. for 문을 이용한 정렬 문제예요.

[기출 예제] 25년 1회 실기 (난이도: ★★☆)

다음은 C언어에 대한 문제이다. 아래 코드를 확인하여 알맞는 출력값을 작성하시오.

▶ 정답
4
BACDE

```
1  #include <stdio.h>
2  char Data[5] = {'B', 'A', 'D', 'E'};
3  char c;
4
5  int main(){
6      int i, temp, temp2;
7
8      c = 'C';
9      printf("%d\n", Data[3]-Data[1]);
10
11     for(i=0;i<5;++i){
12         if(Data[i]>c)
13             break;
14     }
15
16     temp = Data[i];
17     Data[i] = c;
18     i++;
19
20     for(;i<5;++i){
21         temp2 = Data[i];
22         Data[i] = temp;
23         temp = temp2;
24     }
25
26     for(i=0;i<5;i++){
27         printf("%c", Data[i]);
28     }
29 }
```

Data[3]은 E 이고, Data[1]은 A에요.
사실, 각각의 ASCII 코드는 69, 65 이기 때문에 69-65=4가 나와요.
하지만, ASCII 코드 값을 몰라도 풀 수 있어요.
A < B < C … < Z 순으로 +1씩 커진다는 걸 알고 있으면 돼요.
A,B,C,D,E 총 4 단계 차이 때문에 4가 출력됨을 알 수 있어요.

B, A, D, E 순으로 순회를 하는데,
C 보다 큰 (뒤에 위치한) 문자가 나오면 빠져나와요.
C 보다 뒤에 위치한 **문자의 인덱스 i를 구하기** 위한 for 문이에요.

스왑하기 위해 temp 변수에 기존 Data[i] = Data[3] = E 를 할당해요.
E 자리에는 문자 C를 할당해요.
i를 +1 해서 3가 돼요.

여기도 스왑하기 위한 for 문이에요.
i가 3부터 Data[3]을 temp2에 임시저장
Data[3] 자리에 위에서 할당한 temp E를 할당
temp 에는 [21행]에서 저장한 E를 할당해요.

i가 4인 경우도 동일하게 반복하면 B,A,C,D,E 로 배열이 정렬돼요.
[27행]은 배열을 출력하라는 구문이기에, BACDE 가 출력돼요.

정답은
4
BACDE 가 돼요.

[문제 유형] 연산자

[기출 예제] 23년 2회 실기 (난이도: ★★☆)

▶ 정답 n[(i+1) % 5]

다음 C언어로 구현된 프로그램을 실행시킨 결과가 "43215"일 때, <처리조건>을 참고하여 괄호에 들어갈 알맞은 식을 쓰시오.
입력값이 54321일 경우 출력값이 43215로 출력되어야 한다.
<처리조건>

★ 모듈로(modulo) 연산자에 대한 설명은 위 페이지를 참고해주세요!

괄호의 식에 사용할 문자는 다음으로 제한한다.
- n, i
- +, -, /, *, %
- 0~9, (,), [,]

```
1  #include <stdio.h>
2
3  int main(void) {
4      int n[5];
5      int i;
6
7      for (i = 0; i < 5; i++) {
8          printf("숫자를 입력해주세요 : ");
9          scanf("%d", &n[i]);
10     }
11
12     for (i = 0; i < 5; i++) {
13         printf("%d", (              ) );
14     }
15
16     return 0;
17 }
```

n이라는 이름의 정수 배열을 선언했어요. 크기는 5예요.
i라는 정수 변수도 선언했어요. 이건 반복문에서 사용할 거에요.

for 반복문을 사용해서 사용자로부터 5개의 숫자를 입력받아요.
반복할 때마다 "숫자를 입력해주세요 : "라는 메시지를 출력해요.
scanf 함수를 사용해서 사용자가 입력한 숫자를 n 배열에 저장해요.

다른 for 반복문을 사용해서 n 배열에 저장된 숫자들을 출력해요.
이때, 괄호 안에는 우리가 출력하고 싶은 순서대로 숫자를 가져와야 해요.

n[0]	n[1]	n[2]	n[3]	n[4]
5	4	3	2	1

위의 배열처럼 값이 저장되어 있고, 우리는 43215를 출력해야해요.
이 순서를 잘 보면, 인덱스가 1, 2, 3, 4, 0 순으로 변하고 있어요.

첫 번째 인덱스는 1이고,
다음 인덱스는 이전 인덱스에 1을 더한 값이에요.
그런데 인덱스가 4 다음에는 0으로 "회전"하고 있어요.

이걸 코드로 표현하면 (i+1) % 5가 되는 거예요.

i는 0부터 4까지 변하는 반복문의 변수예요.
i+1은 1부터 5까지 변하겠죠.
여기에 % 5를 하면, 값의 범위는 0부터 4가 돼요.

(0+1) % 5 = 1
(1+1) % 5 = 2
(2+1) % 5 = 3
(3+1) % 5 = 4
(4+1) % 5 = 0

[기출 예제] 23년 1회 실기 (난이도: ★★☆)

다음은 2진수 101110을 10진수로 변환하는 C언어 프로그램이다. 프로그램을 분석하여 괄호(a~b)에 들어갈 알맞은 답을 쓰시오.

▶ 정답
a. %
b. 10

```c
#include <stdio.h>

int main() {

    int input = 101110;
    int di = 1;
    int sum = 0;

    while (1) {

        if (input == 0) {
            break
        } else {

            sum = sum + (input (a)(b)) * di;
            di = di * 2;
            input = input / 10;

        }
    }

    printf("%d", sum);

    return 0;
}
```

2진수 101110을 10진수로 변환하고자 하는 입력값 input을 정의하고 있어요.
2진수의 각 자리수를 10진수로 변환할 때 사용할 배수 di를 정의합니다.
2진수의 가장 오른쪽 자리부터 시작하여, 2의 거듭제곱을 나타내는 값이에요.
변환된 10진수 값을 누적할 변수 sum을 0으로 할당해요.

무한 루프를 시작합니다.

input이 0이 되면 모든 자리를 처리했으므로 반복문을 빠져나옵니다.

이 부분이 중요해요.
여기서는 input의 가장 오른쪽 자리수를 가져와야 합니다.
이를 위해 input을 10으로 나눈 나머지를 사용하면
2진수의 가장 오른쪽 숫자를 얻을 수 있죠.
그래서 괄호 (a)에는 '%'를, (b)에는 '10'을 넣어야 해요.

di를 2배 증가시키는데, 이는 2진수의 다음 자리가 10진수로
2배 더 큰 가치를 가지기 때문이에요.

input을 10으로 나눠서 다음 자리수로 넘어갑니다.
이는 2진수를 한 자리씩 오른쪽으로 이동시키는 효과가 있죠.

예를 들어, 101110 을 10으로 나누면, 10111 이 되면서 한 자리 오른쪽 이동이 되었어요.

★ 2진수 계산은 이렇게 할 수 있어요

곱해요

2^3	2^2	2^1	2^0
1	0	1	1

2진수를 계산할 때, 각 자리수의 2의 제곱 * 값(1 또는 0)이에요.
예에서는 $(2^3 \times 1) + (2^2 \times 1) + (2^1 \times 1) + (2^0 \times 1)$ 으로 표현할 수 있고, 1011의 10진수는 8+0+2+1 = 11이 돼요.

★ 연산자 종류

관계	==	같다	>	크다	<	작다
	!=	같지 않다	>=	크거나 같다	<=	작거나 같다
비트	&	and	모든 비트가 1이면 1			
	^	xor	모든 비트가 같으면 0, 하나라도 다르면 1			
	|	or	모든 비트 중 하나라도 1이면 1			
	~	not	각 비트의 부정 (0이면 1 산출)			
	<<	-	비트를 왼쪽으로 이동			
	>>	-	비트를 오른쪽으로 이동			
논리	!	not	&&	and	||	or

★ 연산자는 외워두면 좋아요

왼쪽 표를 먼저 숙지하고, 문제를 푸는 게 좋아요.
외울 게 많아 보이지만, 현실에서의 수학 기호와 비슷해서 외우는데 크게 어렵지 않을 거 에요!

[기출 예제] 22년 1회 실기 (난이도: ★★☆)

다음은 정수를 역순으로 출력하는 C언어 프로그램이다. 예를 들어 1234의 역순은 4321이다.
단, 1230 처럼 0으로 끝나는 정수는 고려하지 않는다. 프로그램을 분석하여 괄호(①~③)에 들어갈 알맞은 연산자를 쓰시오.

▶ 정답
① >
② %
③ /

```c
#include <stdio.h>

int main() {

    int number = 1234;
    int div = 10;
    int result = 0;

    while (number ( ① ) 0) {
        result = result * div;
        result = result + number ( ② ) div;
        number = number ( ③ ) div;
    }

    printf("%d", result);
    return 0;

}
```

number = 1234; → 뒤집을 숫자가 저장되어 있어요 (예: 1234).
int div = 10; → 10으로 설정돼 있어요. 이건 나중에 숫자를 자릿수별로 나누는 데 사용돼요.
int result = 0; → 최종 결과를 저장하기 위해 0으로 할당되어 있어요.
while (number (①) 0) { → while 반복문을 사용해서 number가 0이 될 때까지 반복해요.
result = result * div; → result는 10배(div) 증가시키고 (result * div)
result = result + number (②) div; → number의 마지막 자릿수를 추가해요 (number % div).
number = number (③) div; → 10으로 나눠서 한 자리씩 줄여나가요 (number / div).

★ 이런 식으로 유추할 수 있어요.

먼저, 주어진 정보는 [10행] 에서 result에 10을 곱한다는 거에요.
이 말은 1이라면 10이 될테고, 25라면 250이 될 거에요.
즉, 왼쪽으로 한 칸 이동시키겠다는 << 비트연산자와 유사해요.

그럼 처음 result에 들어간 값을 계속 왼쪽으로 이동한다는 의미이고,
우리의 number 1234 중에서는 4가 result에 먼저 들어가야 계속해서 왼쪽으로 이동해서 결국에는 역순이 될 수 있겠죠?

number (②) div = 4가 나오려면, % 연산자를 사용해야 해요.
숫자를 대입하면, 1234 (②) 10 = 4인 건데, 10으로 나눈 나머지가 필요하기 때문이에요.

두 번째로, number에는 4는 필요없어졌으니, number의 4를 지울 필요가 있겠죠?
즉, number를 오른쪽 이동 시켜야 하는데, 이때는 10을 나눠주면 돼요.
숫자를 대입하면, 1234 (③) 10 = 123이 돼야하는데, 그래서 / 연산자가 필요해요.

마지막으로, (①)를 찾아야 해요.
[12행]에서 number가 계속 10씩 나누기 한다는 사실을 알았고,
0이 되면, result에 넣을 number 값도 없겠죠.
즉, 자연스럽게 (①) 은 > 이 돼요.
number가 0 보다는 커야 역순 처리를 할 값이 존재할테니까요!

★ 삼항 연산자
삼항 연산자 ?는 조건식 ? 참일 때 값 : 거짓일 때 값 형태로 동작합니다.

True ? "a" : "b" -> a 반환
False ? "c" : "d" -> d 반환

[기출 예제] 24년 1회 실기 (난이도: ★★☆)

다음 C언어 코드에서 알맞는 출력 값을 작성하시오.

▶ 정답
151

```c
#include <stdio.h>

int main() {
    int v1 = 0, v2 = 35, v3 = 29;

    if(v1 > v2 ? v2 : v1) {
        v2 = v2 << 2;
    } else {
        v3 = v3 << 2;
    }

    printf("%d", v2+v3);
}
```

세 개의 정수형 변수 v1, v2, v3가 선언되고 각각 0, 35, 29로 초기화돼요.

if(v1 > v2 ? v2 : v1) 는
if(0 > 35 ? 35 : 0) 으로 해석되고,
if(false ? 35: 0) 이므로,
if(0) 이 됩니다.

if 문 안에 0이 들어가면 거짓이기 때문에 else 가 수행되게 됩니다.

v3 = v3 << 2; 는 비트 연산을 수행합니다.
v3를 왼쪽으로 두 비트 시프트합니다.
V3는 29이고, 2진수로 11101입니다.
왼쪽으로 2칸 이동하면 1110100 이고
1110100 은 10진수로 116입니다.
116을 v3에 덮어쓰게 됩니다.

[12행] 은 v2 + v3 = 35 + 116 = 151 이 출력되게 됩니다.

삼항 연산자, 비트 연산자의 개념을 모르면 풀기 어려울 수 있는 문제에요.

하지만, 한번만 이해했다면 다시 만나면 쉽게 풀 수 있어요!

★ Key Point
구조체와 포인터, 그리고 비트 AND(&) 연산, 16진수에 대해 이해하고 있어야 해요.
개념만 알면 쉽지만, 이전에 나오지 않은 스타일이기에 어려울 수 있어요.

- 비트 AND 연산 (&): dec 함수 내의 enc & 0xA5는 enc와 0xA5의 각 비트를 비교하여, 두 비트가 모두 1인 경우에만 1을 반환합니다.
- 16진수: 코드에 0xA0, 0xA5와 같이 0x로 시작하는 값은 16진수를 의미합니다. 비트 연산을 위해서는 이를 2진수로 변환하여 계산해야 합니다.

[기출 예제] 25년 1회 실기 (난이도: ★★★)

다음 C언어 코드에서 알맞는 출력 값을 작성하시오.

▶ 정답
908

```c
#include <stdio.h>

typedef struct student {
    char* name;
    int score[3];
} Student;

int dec(int enc) {
    return enc & 0xA5;
}

int sum(Student* p) {
    return dec(p->score[0]) + dec(p->score[1]) + dec(p->score[2]);
}

int main() {
    Student s[2] = { "Kim", {0xA0, 0xA5, 0xDB}, "Lee", {0xA0, 0xED, 0x81} };
    Student* p = s;
    int result = 0;

    for (int i = 0; i < 2; i++) {
        result += sum(&s[i]);
    }
    printf("%d", result);
    return 0;
}
```

Student 구조체를 정의해요.

학생 이름을 가리킬 포인터
3개의 정수형 점수를 저장할 배열

★ 0xA5 2진수
16진수는 1,2,3,4,5,6,7,8,9,A(10),B(11),C(12),D(13),E(14),F(15) 로 표현해요.
A는 10을, 5는 5을 의미해요. 2진수로 표현하면 1010, 0101 이에요.
합쳐서 10100101 이에요.

정수(enc)를 받아 0xA5 (10100101)와 비트 AND 연산을 한 결과를 반환해요.
enc를 2진수로 변환하고, 같은 자리 수의 비트가 모두 1이어야 1을 반환, 그 외에는 0을 반환해요.

Kim

0xA0 (2진수: 10100000)	0xA5 (2진수: 10100101)	0xDB (2진수: 11011011)
0xA5 (2진수: 10100101)	0xA5 (2진수: 10100101)	0xA5 (2진수: 10100101)
10100000 (10진수: 160)	10100101 (10진수: 165)	10000001 (10진수: 129)

Lee

0xA0 (2진수: 10100000)	0xED (2진수: 11101101)	0x81 (2진수: 10000001)
0xA5 (2진수: 10100101)	0xA5 (2진수: 10100101)	0xA5 (2진수: 10100101)
10100000 (10진수: 160)	10100101 (10진수: 165)	10000001 (10진수: 129)

160 + 165 + 129 + 160 + 165 + 129 = 908 이 result에 저장돼요.

908이 출력돼요.

[문제 유형] 단순 while문

★ while 구문

while 구문은 '조건을 만족하는 동안 계속 반복해' 라는 뜻이에요. 마치 "이 책 재미있으면 계속 읽어"라고 말하는 것처럼, 프로그램에게 어떤 조건이 참인 동안은 특정한 일을 계속 하라고 지시하는 거죠.

```
int i = 0; // 'i'를 0으로 시작해요.
while (i < 5) { // 'i'가 5보다 작은 동안에는, 아래 명령을 반복해요.
   printf("%d₩n", i); // 'i'의 값을 화면에 보여줘요.
   i++; // 'i'를 1씩 증가시켜요. 이걸 '증감'이라고 해요.
}
```

이 프로그램은 'i'를 0에서 시작해서 1씩 증가시키면서, 'i'의 값이 5보다 작을 때까지 'i'의 값을 화면에 출력해요. 'i'가 5가 되면, 조건 'i < 5'가 거짓이 되고, 그래서 'while' 반복문을 끝내게 됩니다. 그래서 화면에는 0부터 4까지의 숫자가 차례로 보여지겠죠.

[기출 예제] 20년 3회 실기 (난이도: ★☆☆)

▶ 정답 0

다음 C언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오.
(단, 출력문의 출력 서식을 준수하시오.)

```
1  #include <stdio.h>
2
3  void main() {
4      int i = 0, c = 0; ⟶ i와 c라는 두 개의 정수 변수가 선언되어 있고, 둘 다 0으로 할당되어 있어요.
5      while (i < 10) { ⟶ i가 10보다 작을 때까지 계속 반복됩니다.
6          i++; ⟶ 반복문 내에서, 먼저 i를 1씩 증가시켜요 (i++).
7          c*=i; ⟶ c에 i를 곱한 값을 다시 c에 할당해요 (c *= i). 이것은 c = c * i와 같은 의미에요.
8      }
9      printf("%d",c); ⟶ 정답은 0이에요.
10 }
```

★ 0에다 어떤 숫자를 곱해도 0이에요

i = 1
c = c * i = **0** * 1 = 0
i = 2
c = c * i = **0** * 2 = 0
i = 3
c = c * i = **0** * 3 = 0
i = 4
c = c * i = **0** * 4 = 0
i = 5
c = c * i = **0** * 5 = 0
i = 6
c = c * i = **0** * 6 = 0
i = 7
c = c * i = **0** * 7 = 0
i = 8
c = c * i = **0** * 8 = 0
i = 9
c = c * i = **0** * 9 = 0

[문제 유형] 재귀 함수

★ 재귀 함수
함수 내부에서 자기 자신을 다시 호출하는 함수예요. 여기서 f(n)은 n과 f(n-1)의 곱으로 정의되고 있어요.
재귀 함수를 풀 때는, [4행] 처럼 종료조건(더 이상 재귀하지 않고 반환하는 조건) 까지 함수를 나열했다가 거꾸로 값을 계산하면서 최초 함수까지 올라오면 돼요.
이런 식으로 함수가 자기 자신을 호출하면서 문제를 더 작은 문제로 나누어 풀어요.

[기출 예제] 23년 3회 실기 (난이도: ★☆☆)

다음 C 프로그램 출력 결과 구하기

▶ 정답 5040

```
1  #include <stdio.h>
2
3  int f(int n) {
4      if(n<=1) return 1;
5      else return n*f(n-1);
6  }
7
8  int main() {
9      printf("%d", f(7));
10 }
```

정수 n을 매개변수로 받아요.
만약 n이 1보다 작거나 같으면 1을 반환해요.
그렇지 않으면 n과 f(n-1)의 곱을 반환해요.

이 함수는 재귀 함수라고 불려요.

f(7)을 호출하고 그 결과를 출력하고 있어요.

f(7) = 7 x f(6)
= 7 x 6 x f(5)
= 7 x 6 x 5 x f(4)
= 7 x 6 x 5 x 4 x f(3)
= 7 x 6 x 5 x 4 x 3 x f(2)
= 7 x 6 x 5 x 4 x 3 x 2 x f(1)
= 7 x 6 x 5 x 4 x 3 x 2 x 1

즉, 수학에서의 팩토리얼 계산을 구현한 것이에요.
팩토리얼은 1부터 어떤 양의 정수까지의 모든 양의 정수의 곱이에요.
예를 들어, 7!은 1 * 2 * 3 * 4 * 5 * 6 * 7 = 5040이 되는 거죠.

그래서 정답은 5040이 돼요!

[기출 예제] 22년 1회 실기 (난이도: ★☆☆)

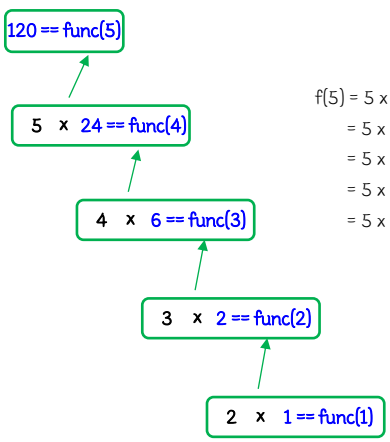
다음 C언어로 구현된 프로그램을 분석하여 5를 입력했을 때 그 실행 결과를 쓰시오.
(단, 출력문의 출력 서식을 준수하시오.)

▶ 정답 120

```
1  #include <stdio.h>
2
3  int func(int a) {
4    if (a <= 1) return 1;
5    return a * func(a - 1);
6  }
7
8  int main() {
9    int a;
10   scanf("%d", &a);
11   printf("%d", func(a));
12 }
```

하나의 정수 a를 매개변수로 받아요.
a가 1 이하일 경우 1을 반환해요. (종료 조건)
a에 func(a - 1)의 결과를 곱해서 반환해요.
이 부분이 재귀 호출이에요.

사용자로부터 정수 a를 입력받아요.
func(a)를 호출해서 그 결과를 출력해요.
여기서는 5를 넣어서 호출할거에요
정답은 5! = 5*4*3*2*1 = 120이에요.



f(5) = 5 x f(4)
= 5 x 4 x f(3)
= 5 x 4 x 3 x f(2)
= 5 x 4 x 3 x 2 x f(1)
= 5 x 4 x 3 x 2 x 1

[문제 유형] 정렬

[기출 예제] 20년 1회 실기 (난이도: ★★☆)

▶ 정답 50 75 85 95 100

다음 C언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오.
(단, 출력문의 출력 서식을 준수하시오.)

★ key point
버블 정렬이든, 선택 정렬이든 정렬 종료에 상관없이 정렬 문제인걸 캐치 했다면, 오름차순인지 내림차순인지만 정확하게 파악하고 순서대로 출력하면 돼요!
꼭 손으로 다 할 필요는 없어요!

```c
#include <stdio.h>

void main() {
    int i,j;
    int temp;
    int a[5] = { 75, 100, 95, 50, 85 };

    for(i=0; i<4; i++){
        for(j=0; j<4-i; j++){
            if(a[j] > a[j+1]){
                temp=a[j];
                a[j] = a[j+1];
                a[j+1] = temp;
            }
        }
    }

    for(i=0; i<5; i++){
        printf("%d ", a[i]);
    }
}
```

a[5]라는 정수 배열이 선언되어 있고, [75, 100, 95, 50, 85] 5개의 요소로 할당되어 있어요

처음부터 끝에서 -1까지 (버블정렬이 완료된 배열 수를 뜻해요)

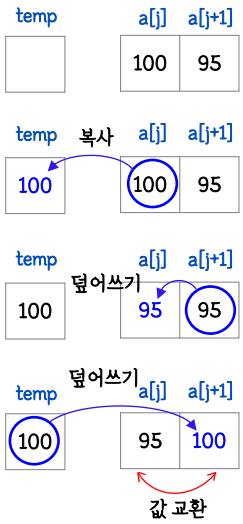
처음부터 (끝 -1 -버블정렬 완료된 수) 까지

현재가 다음 배열보다 크면 바꿔요.

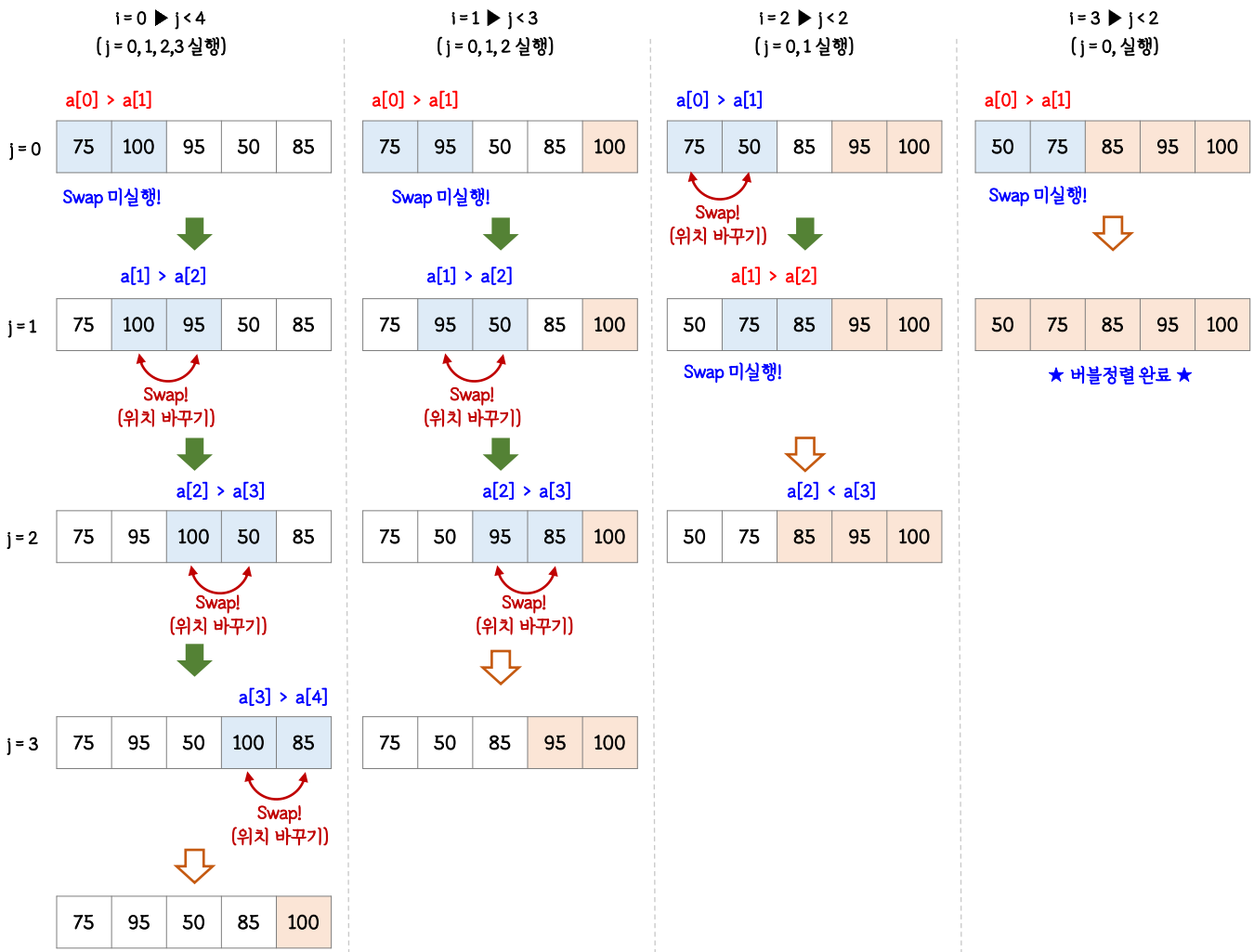
swap

정렬된 배열을 출력해요 정답은 50 75 85 95 100 이에요.

★ swap
[11행]의 로직 동작 과정이에요.



★ 버블 정렬



[기출 예제] 23년 1회 실기 (난이도: ★★☆)

다음은 버블 정렬을 이용하여 배열에 저장된 수를 오름차순으로 정렬하는 프로그램이다. 프로그램을 분석하여 괄호(①, ②)에 들어갈 알맞은 답을 쓰시오.

▶ 정답
① idx2
② nx

```c
#include <stdio.h>

void swap(int* a, int idx1, int idx2) {
    int t = a[idx1];
    a[idx1] = a[idx2];
    a[ ① ] = t;
}

void sort(int* a, int len) {
    for (int i = 0; i < len-1; i++)
        for (int j = 0; j < len-1-i; j++)
            if (a[j] > a[j+1])
                swap(a, j, j+1);
}

void main() {
    int a[] = { 85, 75, 50, 100, 95 };
    int nx = 5;
    sort(a, ② );
}
```

- 배열의 idx1 위치에 있는 원소를 임시 변수 t에 저장합니다.
- idx2 위치에 있는 원소를 idx1 위치로 이동합니다.
- 처음 idx1 위치에 있던 원소를 idx2 위치로 이동합니다. 여기서 괄호 (①)에는 idx2가 들어가야 합니다.
- 배열 a를를 버블 정렬 방식으로 정렬합니다. len은 배열의 길이입니다.
- 바깥쪽 for 반복문은 정렬할 배열의 길이만큼 반복합니다. 배열의 모든 원소를 확인해야 하니까요.
- 안쪽 for 반복문은 정렬된 원소를 제외하고 나머지 원소들을 비교합니다.
- 인접한 두 원소의 크기를 비교하고, 앞쪽 원소가 더 크면 위치를 바꿉니다.
- [3행] swap 함수를 호출하여 j번째와 j+1번째 원소의 위치를 바꿉니다.
- 정렬할 배열을 할당합니다.
- 배열 a의 원소 개수를 저장하는 변수입니다.
- [9행] sort 함수를 호출하여 배열 a를 정렬합니다. 여기서 괄호 (②)에는 배열의 길이인 nx가 들어가야 합니다.

★ key point
여기서는 오름차순 정렬이라는 힌트를 문제에서 줬어요.
메소드명 swap이고, swap은 일반적으로 두 요소의 위치를 바꾸는 걸 의미해요. 우리는 그럼 두 요소의 위치를 바꾸는 코드를 작성해주면 될 거라는 걸 유추해볼 수 있어요.

idx1의 값을 t라는 변수에 임시 저장했고, idx1 요소는 idx2로 덮어썼고 idx2의 요소에 idx1을 넣어야 서로 위치가 바뀌는 효과가 있겠죠?
그래서, ①에는 idx2가 들어갸요.

너무 어렵다면, 바로 위 페이지의 ★ swap을 참고해주세요!

★ key point
원래 정상적인 문제 풀이의 흐름으로는 배열의 끝까지 반복해야하니 ②의 값은 nx가 된다고 생각하는게 올바른 풀이에요.

하지만, **실제 시험장에서 코드 해석이 어려워 찍어야 하는 상황에서는 다음처럼 유추해서 정답을 맞출 수도 있어요.** sort에 들어갈 수 있는건 main문의 변수가 a 배열과 nx 두개만 존재하는 상황에 첫번째 매개변수로 a가 이미 들어간 상황에서는 추가로 넣어줄 변수는 nx밖에 없어 nx가 정답이 될 거라는 생각도 해볼 수 있어요. 이 방법은 답을 아예 모르는 상황에서 조금이라도 정답에 가까운 답을 적기 위한 방법이지 절대 옳은 방법이 아니에요!

[기출 예제] 23년 2회 실기 (난이도: ★★☆)

다음은 데이터를 오름차순으로 정렬하는 선택 정렬 알고리즘을 C 언어 프로그램으로 구현한 것이다. 프로그램을 분석하여 괄호에 들어갈 알맞은 연산자를 쓰시오.

<처리조건>

괄호의 식에 사용할 문자는 다음으로 제한한다.
- +=, -=, *=, /=
- ==, !=, >, >=, <, <=
- &&, ||

▶ 정답
① >

★ key point
정렬 문제에서 각 요소들의 정렬 조건에 들어갈 빈칸이니, 두 요소의 비교가 크거나(<), 작거나(>), 크거나 같거나(<=), 작거나 같거나(>=) 만 들어갈 수 있음을 유추할 수 있어요.
정렬 종류에 상관없이 현재 오름 차순으로 정렬하려고 하고, 작은 순으로 정렬이 되어야할 때, 왼쪽의 요소가 오른쪽 요소보다 클 때 바꿔줘야 작은 순으로 정렬될 수 있어요.
j는 i보다 +1이 크기때문에, i는 왼쪽 요소 j는 오른쪽 요소에요. 왼쪽이 오른쪽보다 크면 두 요소를 바꿔야하니 >이 돼요. (>=는 왜 정답이 아니냐면, 틀린 방법은 아니지만 같을 때 굳이 두 요소를 변경할 필요가 없기 때문이에요)

```c
#include <stdio.h>
int main() {
    int E[] = {64, 25, 12, 22, 11};
    int n = sizeof(E) / sizeof(E[0]);
    int i = 0;
    do {
        int j = i + 1;
        do {
            if (E[i] (  ①  ) E[j]) {
                int tmp = E[i];
                E[i] = E[j];
                E[j] = tmp;
            }
            j++;
        } while (j < n);
        i++;
    } while (i < n-1);
    for(int i=0; i<=4; i++)
        printf("%d ", E[i]);
}
```

- 정렬할 데이터가 저장된 배열 E를 선언하고 할당하고 있어요.
- 배열 E의 전체 크기를 첫 번째 요소의 크기로 나눠서 배열에 몇 개의 요소가 있는지를 계산해요. 즉, 배열 E의 길이 n을 구하는 거죠.
- 내부 루프에서 비교할 두 번째 요소의 시작점을 설정해요. i 다음 요소부터 시작하죠.
- 이 부분이 중요한데요, 여기서는 E[i]가 E[j]보다 클 때 두 요소의 위치를 바꿔줘야 해요. 오름차순 정렬을 하려면 현재 선택된 요소(E[i])가 비교 대상 요소(E[j])보다 클 경우에만 위치를 바꾸면 되거든요.

  따라서 괄호에 들어갈 알맞은 연산자는 > 입니다.
  이 조건은 '만약 E[i]가 E[j]보다 크다면'을 의미하죠.
- i가 배열의 길이보다 하나 작을 때까지 반복해요.

[기출 예제] 24년 1회 실기 (난이도: ★★☆)

다음은 C언어에 대한 문제이다. 알맞는 출력 값을 작성하시오.

이 문제는 문자열을 뒤집고 나서 특정 위치의 문자들을 출력하는 문제에요.
문제를 풀 때, main 을 먼저 살펴보세요.

▶ 정답
GECA

```c
#include <stdio.h>
#include <string.h>

void reverse(char* str){
    int len = strlen(str);
    char* p1 = str;
    char* p2 = str + len - 1;
    while(p1<p2){
        char t = *p1;
        *p1 = *p2;
        *p2 = t;
        p1++;
        p2--;
    }
}

int main() {
    char str[100] = "ABCDEFGH";

    reverse(str);

    int len = strlen(str);

    for(int i=1; i<len; i+=2) {
        printf("%c",str[i]);
    }

    printf("\n");

    return 0;

}
```

strlen(str)을 사용하여 문자열의 길이를 구해요.

p1은 문자열의 시작을,
p2는 문자열의 끝을 가리켜요.

p1의 값을 t에 임시로 옮기고,
*p1에 *p2의 값을 넣고,
*p2에 임시로 넣어둔 t의 값을 넣어요. (이러면 양끝의 값이 바뀌어요)
p1++과 p2--를 통해, 양 극단에서 한 칸씩 가까워지면서
그 두 값을 바꾸고 p1이 p2보다 같거나 커지면 그만둬요.

결과적으로는, 파라미터 str의 값이 뒤집어져서
HGFEDCBA가 돼요.

str 배열을 선언하고 "ABCDEFGH"로 초기화하고 있어요.

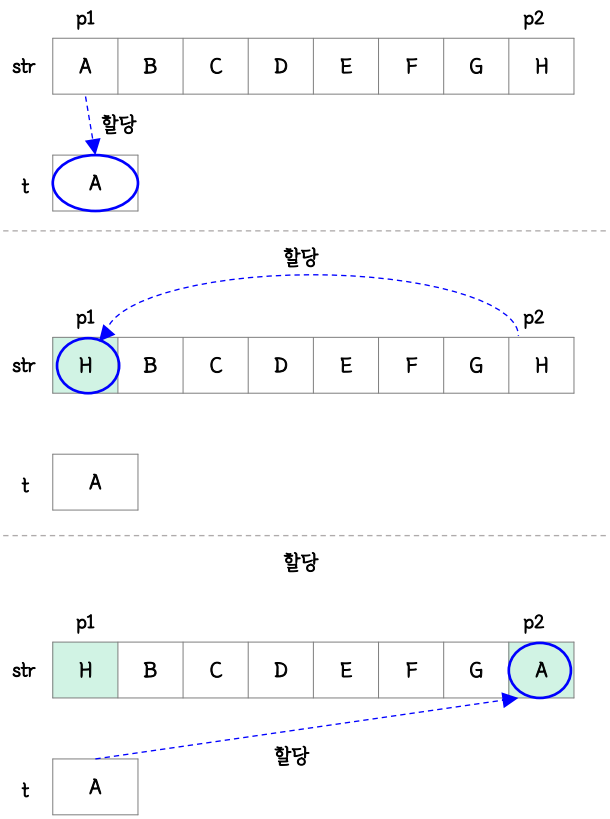
reverse 함수를 호출하여 str에 대해서 거꾸로 뒤집어요.

str의 길이를 len에 할당했어요.

i가 0 이 아닌 1 부터 2씩 증가하면서 len(끝)까지 반복해요.

i 번째 index 값을 출력해요. len은 8이므로, 1부터 8미만까지 동작해요.
1 -> 3 -> 5 -> 7 index의 문자를 출력해요.
str은 HGFEDCBA 이므로, GECA가 정답이 돼요.

줄내림을 해요.



[기출 예제] 24년 2회 실기 (난이도: ★★★) 오래 걸리지는 않지만, 실수하기 좋은 문제에요.

다음은 C언어에 대한 문제이다. 아래 코드를 확인하여 알맞는 출력 값을 작성하시오.

▶ 정답
-13

```c
#include <stdio.h>

void swap(int a, int b) {
    int t = a;
    a = b;
    b = t;
}

int main() {

    int a = 11;
    int b = 19;
    swap(a, b);

    switch(a) {
        case 1:
            b += 1;
        case 11:
            b += 2;
        default:
            b += 3;
        break;
    }

    printf("%d", a-b);
}
```

C에서 기본적으로 함수 매개변수는 '값에 의한 전달' 방식을 사용해요.
이는 함수 내에서 매개변수 값을 변경해도 호출한 곳의 원래 변수에는 영향을 주지 않는다는 뜻이에요.
포인터로 주소를 전달하지 않는 한 swap 함수가 실제로 a와 b의 값을 바꾸지 않아요.

a와 b라는 두 개의 정수 변수를 만들고 각각 11과 19로 초기화해요.

swap 함수를 호출해요. 하지만 이 호출은 위에서 설명한 것처럼 a와 b의 값을 실제로 바꾸지 않아요.

a가 11이므로 case 11부터 시작해서 b에 2를 더하고,
그 다음 default로 넘어가 b에 3을 더 더해요.
break; 가 없기 때문이에요.
b는 24가 돼요.

11 – 24 이므로 -13 이 출력돼요.

★ key point
swap 방식은 꾸준히 출제된 유형의 문제에요.
그래서, 포인터의 개념을 아는 지 출제 의도를 넣어 꼬아서 낸 문제에요.
단순히, swap 방식으로 문제를 풀었다면 틀렸을 거에요.

[문제 유형] 포인터

★ 포인터

포인터는 매우 중요하지만 어려울 수 있는 개념이에요. 포인터는 메모리 주소를 저장하는 변수로, 특정한 데이터의 위치(주소)를 가리킵니다. 이를 통해 직접적으로 메모리에 접근하고, 데이터를 효율적으로 관리할 수 있습니다. P = &a;라고 할당하면, p는 변수 a의 주소를 가리키게 됩니다. *p를 사용하면 p가 가리키는 주소에 있는 데이터, 즉 a의 값에 접근할 수 있습니다.

포인터 문제는 정말 많이 반복해서 출제되지만, 문제의 형식을 계속 유사하게 출제됩니다. **포인터 개념을 이해하고 가면 정말 좋지만 너무 어렵다면 문제의 유형별 답을 찾는 방법만 암기하시는 것도 방법**이에요!

[기출 예제] 23년 1회 실기 (난이도: ★★☆)

다음 C언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오.

▶ 정답
Art
A
A
Art
Art

```
1  #include <stdio.h>
2
3  int main(){
4
5      char a[] = "Art";
6      char* p = NULL;
7      p = a;
8
9      printf("%s\n", a);
10     printf("%c\n", *p);
11     printf("%c\n", *a);
12     printf("%s\n", p);
13
14     for(int i = 0; a[i] != '\0'; i++)
15        printf("%c", a[i]);
16
17 }
```

역슬래시n은 Enter와 비슷해요. 한 줄 내림해요.

배열에 바로 문자열을 저장하는 경우, 그 끝을 알 수 없기 때문에 역슬래시 0(= null = ₩0 = \0 전부 같은 표현이에요 운영체제나 환경별로 표시되는 형식만 다를뿐이에요.) 이 배열의 끝에 추가돼요.

문자열 "Art"를 저장하는 문자 배열 a를 선언하고 할당하고 있어요.

이 배열에는 'A', 'r', 't', 그리고 문자열의 끝을 나타내는 널 문자('₩0')까지 총 4개의 문자가 들어 있죠.

p라는 이름의 포인터를 선언하고 NULL로 할당해요.

포인터 p에 배열 a의 주소를 할당해요. 이제 p는 문자열 "Art"가 저장된 배열 a를 가리키고 있죠.

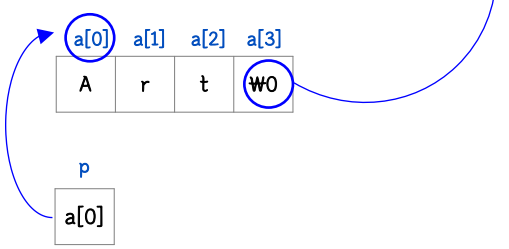
문자열 a를 출력하는 코드예요. %s는 문자열을 출력하는 형식 지정자이며, "Art"가 출력됩니다.

p가 가리키는 위치의 문자를 출력해요. %c는 단일 문자를 출력하는 형식 지정자이며, a의 첫 번째 요소를 가리키므로 'A'가 출력됩니다.

배열 이름 a를 포인터처럼 사용해서 첫 번째 문자를 출력해요. 여기서도 'A'가 출력돼요.

포인터 p가 가리키는 문자열을 출력해요. p는 "Art"를 가리키므로 "Art"가 출력됩니다.

문자열 a의 각 문자를 하나씩 출력하는 반복문이에요.
널 문자('₩0')를 만날 때까지 각 문자를 순서대로 출력합니다.
결과적으로 "Art"가 한 글자씩 출력되죠.



[7행] A라는 값이 아닌 배열 a의 a[0] 메모리 주소를 가리키고 있어요.

★ ASCII 코드

컴퓨터가 문자를 숫자로 표현하는 방법이에요.

문자	10진	문자	10진
A	65	N	78
B	66	O	79
C	67	P	80
D	68	Q	81
E	69	R	82
F	70	S	83
G	71	T	84
H	72	U	85
I	73	V	86
J	74	W	87
K	75	X	88
L	76	Y	89
M	77	Z	90

★ key point

어렵다면, 아래 표현에 대한 정답을 찾는 방법을 외우고 풀어보세요. 계속 반복돼요.

char a[] = "Art"
char* b = "Good"

printf("%s", a) ⟶ "Art"
printf("%c", a[0]) ⟶ "A"
printf("%c", *a) ⟶ "A"
printf("%c", *(a+2)) ⟶ "t"
printf("%s", a+1) ⟶ "rt"

printf("%s", b) ⟶ "Good"
printf("%s", b+1) ⟶ "ood"
printf("%c", *b) ⟶ "G"
printf("%c", *(b+3)) ⟶ "d"

마지막으로 어려울 수 있는데, G에 +3을 하면 J가 돼요. ASCII 코드는 문자를 숫자로 표현하는 방법이에요. A는 65부터 시작해서 Z는 90으로 +1씩 증가해서 매핑되어있어요. *b는 G가 출력돼고, 문자 G의 ASCII 코드는 71이에요. 거기에 +3은 74이고 그래서 J가 출력돼요. 어렵다면, 출력된 문자에 a,b,c,d… 순서대로 옆으로 이동한다 생각해보세요.

가령, *b+3은 G + 3이고, abcd순서대로 나열하면 3칸 옆 알파벳은 G ->H ->I -> J니까 J를 구할 수도 있어요.

printf("%c", *b+3) ⟶ "J"

[기출 예제] 20년 4회 실기 (난이도: ★★☆)

다음 C언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오.
(단, 출력문의 출력 서식을 준수하시오.)

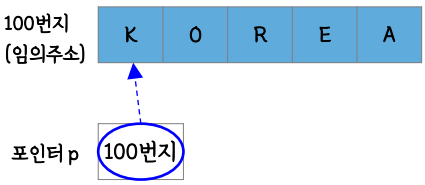
★ 문자열 저장 방법
1. 배열
- ex) char arr[5] = "KOREA"
2. 포인터
- ex) char *p = "KOREA"

▶ 정답
KOREA
EA
K
E
M

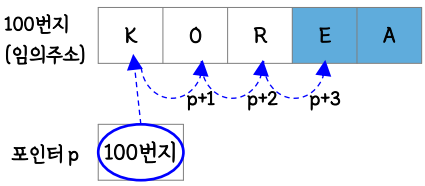
```
1  #include <stdio.h>
2
3  int main(){
4      char *p = "KOREA";
5      printf("%s \n", p);
6      printf("%s \n", p+3);
7      printf("%c \n", *p);
8      printf("%c \n", *(p+3));
9      printf("%c ", *p+2);
10 }
```

4행 → p라는 포인터 변수가 "KOREA"라는 문자열을 가리키고 있어요. 이 때 p는 문자열의 첫 글자인 'K'의 주소를 가지고 있죠.
5행 → p가 가리키는 전체 문자열 "KOREA"를 출력해요.
6행 → p의 주소에서 3만큼 더한 위치, 즉 "KOREA"에서 4번째 글자 'E'부터 시작하는 부분 문자열 "EA"를 출력해요.
7행 → p가 가리키는 위치의 문자, 즉 'K'를 출력해요.
8행 → p에서 3만큼 더한 위치의 문자, 즉 'E'를 출력해요.
9행 → p가 가리키는 문자 'K'의 ASCII 값에 2를 더한 결과를 출력해요.
'K'의 ASCII 값은 75이므로, 75 + 2 = 77이고, 이는 ASCII 테이블에서 'M'에 해당해요.

[4행] printf("%s ₩n", p)
문자열 포인터 p 를 출력

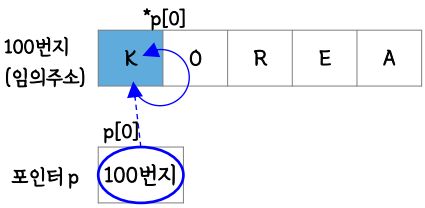


[5행] printf("%s ₩n", p+3);
문자열 포인터 p 에 저장된 값에 3 을 더한 후 출력



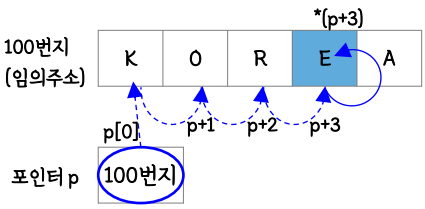
[6행] printf("%c ₩n", *p);
문자열 포인터 p 가 가리키고 있는 주소 영역에 접근하여 값을 가져와서 출력

p 는 p[0]의 주소를 가리키고, 따라서, *p는 *p[0] 이에요.
*(역참조 연산자)는 해당 주소에 존재하는 값을 가져오는 연산자에요.
*p[0]는 p[0] 주소 영역에 존재하는 값을 반환하기 때문에 K만 출력돼요.

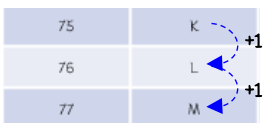


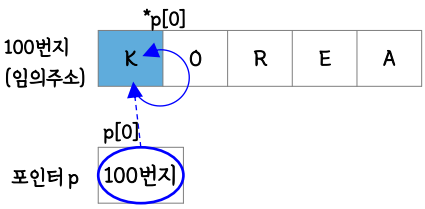
[7행] printf("%c ₩n", *(p+3));
문자열 포인터 p 에 저장된 값에 3을 더한 후, 해당 값이 가리키고 있는 주소 영역에 접근하여 값을 가져와서 출력

p 는 p[0]의 주소를 가리키고, (p+3)은 [5행]과 동일하게 3칸 이동해요.
그 다음, *(역참조 연산자) 를 통해 p+3 주소에 존재하는 값을 반환하기 때문에 E만 출력돼요.



[8행] printf("%s ₩n", *p+2);
*p 의 값(K=75) 을 가져온 후 2를 더하여 출력
위의 아스키 코드를 참고하세요!
A의 아스키코드가 65 인 것만 알아두면 돼요!
▶ 75 + 2 = 77 = M





[기출 예제] 23년 3회 실기 (난이도: ★★☆)

▶ 정답 KOREA OREA K E O

다음 C 프로그램 출력 결과 구하기

★ 중요해요
여러 년도에서 자주 반복 출제되고 있는 문제에요!

```c
#include <stdio.h>

int main() {
    char* p = "KOREA";
    printf("%s ", p);
    printf("%s ", p+1);
    printf("%c ", *p);
    printf("%c ", *(p+3));
    printf("%c ", *p+4);
}
```

- p라는 char 포인터 변수를 선언하고 있어요. 이 포인터는 문자열 "KOREA"의 첫 번째 문자인 'K'의 주소를 가리키고 있어요.
- %s 형식 지정자를 사용하여 p가 가리키는 문자열 전체를 출력해요. 따라서 "KOREA"가 출력될 거에요.
- p+1이 가리키는 문자열을 출력해요. P+1은 p가 가리키는 주소에서 한 칸 옆으로 이동한 주소에요. 즉, 'O'의 주소를 가리켜요. 따라서 "OREA"가 출력될 거에요.
- %c 형식 지정자를 사용하여 *p의 값을 문자로 출력해요. *p는 p가 가리키는 주소에 저장된 값, 즉 'K'예요. 따라서 'K'가 출력될 거에요.
- *(p+3)의 값을 문자로 출력해요. p+3은 p가 가리키는 주소에서 세 칸 옆으로 이동한 주소, 즉 'E'의 주소예요. 따라서 'E'가 출력될 거에요.
- *p+4의 값을 문자로 출력해요. *p는 'K'의 ASCII 코드 값이에요. 거기에 4를 더하면 'O'의 ASCII 코드 값이 돼요. 따라서 'O'가 출력될 거에요.

그래서, 전체 출력되는 정답은 KOREA OREA K E O 이에요.

[기출 예제] 21년 2회 실기 (난이도: ★★☆)

▶ 정답 8

다음 C언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오.
(단, 출력문의 출력 서식을 준수하시오.)

```c
#include <stdio.h>

int main(){

    int ary[3];
    int s = 0;
    *(ary+0) = 1;
    ary[1] = *(ary + 0) + 2;
    ary[2] = *ary + 3;
    for(int i = 0; i < 3; i++) {
        s = s + ary[i];
    }

    printf("%d",s);
}
```

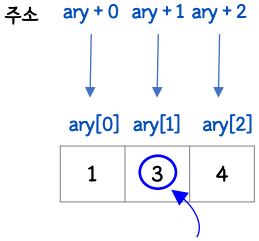
정수 배열에 값을 할당하고, 그 값들을 모두 더해서 출력하는 코드에요.

- ary라는 이름의 정수형 배열이 있어요. 배열 크기는 3이에요.
- s라는 변수는 합계를 저장하기 위해 사용되고, 0으로 할당되어 있어요.
- *(ary+0) = 1;은 ary 배열의 첫 번째 요소(ary[0])에 1을 할당해요.
- 두 번째 요소(ary[1])에 첫 번째 요소의 값(ary[0] 즉, 1)에 2를 더한 값, 즉 3을 할당해요.
- 세 번째 요소(ary[2])에 첫 번째 요소의 값에 3을 더한 값, 즉 4를 할당해요.
- 배열 ary의 모든 요소를 순회하면서 그 값을 변수 s에 더해요.
- s의 최종 값을 출력해요. 정답은 1 + 3 + 4 = 8이에요

★ 배열의 이름은 배열의 0번째 주소에요
ary == &ary[0] 이고, (메모리의 주소)
*ary = ary[0] 이에요. (값)
0번째에 값을 넣고 싶으면,
*ary = 1
ary[0] = 1
*(ary) = 1
다 가능해요.
[7행]에서는 *(ary+0)=1 라고 적혀있는데, *(ary) = 1과 동일한 표현이에요.

★ 배열의 이름에 괄호 없이 더하는 건 포인터 이동이 아니에요.
[8행]과 [9행]의 차이는 무엇일까요?
[8행]은 괄호를 사용했고, [9행]은 괄호 없이 + 3을 했죠.

물론, [8행]도 괄호 안에 0을 더해서 주소의 이동은 없으나, 배열의 0번째 위치에서 N번째 위치의 값을 가져오고 싶을 때는 *(ary + n)으로 표현해야 해요.
*ary + n은 *ary의 값을 가져온 뒤에 정수 n을 더하는 표현이에요.
그러므로, [9행]은 *ary 먼저 가져온 뒤(1), + 3을 진행하여 4 가 나와요.
그 값을 ary[2]에 할당해요.



주소 앞 *을 붙이면 주소의 값을 가져와요
여기서는 *(ary+1)로 표현할 수 있어요

[기출 예제] 21년 3회 실기 (난이도: ★★☆)

▶ 정답 37

다음 C언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오.
(단, 출력문의 출력 서식을 준수하시오.)

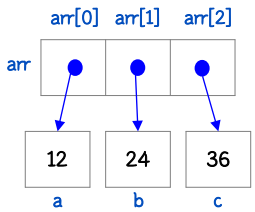
```c
#include <stdio.h>

int main(){
    int *arr[3];
    int a = 12, b = 24, c = 36;

    arr[0] = &a;
    arr[1] = &b;
    arr[2] = &c;

    printf("%d\n", *arr[1] + **arr + 1);
}
```

arr이라는 이름의 포인터 배열이 있어요. 이 배열은 정수에 대한 포인터를 저장할 수 있어요.

a, b, c라는 세 개의 정수 변수가 있어요. 각각 12, 24, 36으로 할당되어 있죠.

각 배열 요소인 arr[0], arr[1], arr[2]는 각각 a, b, c의 메모리 주소를 저장하고 있어요.

*arr[1] + **arr + 1을 계산해서 출력해요.
여기서 *arr[1]은 arr[1]이 가리키는 주소에 있는 값, 즉 b의 값 24를 의미해요.
**arr는 arr[0]이 가리키는 주소에 있는 값, 즉 a의 값 12를 의미해요.
따라서, 이 계산은 24 + 12 + 1이 되겠죠. 이 계산의 결과, 즉 37을 출력해요.



a의 값이 아니라, a의 메모리 주소가 arr[0]에 들어있어요.
즉, arr[0]은 a 메모리 주소고 *arr[0]을 해야 실제 메모리 주소 a에 들어있는 값 12를 가져옵니다.
그러므로 *arr[1]은 24입니다.

배열 이름 arr 자체가 배열의 첫 번째 요소, 즉 arr[0]의 주소를 나타내요.
*arr 은 arr[0]로 표현할 수 있고, arr[0]에 저장된 값을 의미해요.
arr[0]에 저장된 값은 지금 a의 주소를 저장하고 있으니([7행])
한번 더 역참조 연산자(*)로 a에 저장된 값을 가져오기 위해서 **arr(*arr[0])을 사용한 거에요.
그러므로 **arr은 12입니다.

[기출 예제] 23년 1회 실기 (난이도: ★★☆)

▶ 정답 qwe

다음 C언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오.

★ key point
문자열 배열의 끝이 null(\0인 것만 알고 있으면, 크게 어렵지 않은 문제에요.

```c
#include <stdio.h>

int main() {

    char* a = "qwer";
    char* b = "qwtety";

    for(int i = 0; a[i] != '\0' ; i++){
        for(int j = 0; b[j] != '\0'; j++){
            if(a[i] == b[j]) printf("%c", a[i]);
        }
    }

}
```

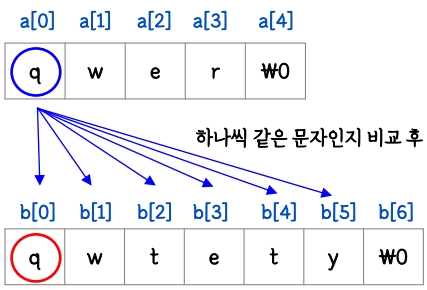
문자열 "qwer"를 가리키는 포인터 a를 선언하고 값을 할당해요.

문자열 "qwtety"를 가리키는 포인터 b를 선언하고 값을 할당해요.

문자열 a를 처음부터 끝까지(널 문자가 나타날 때까지) 확인하는 반복문이에요.
여기서 i는 문자열 a의 각 문자를 순서대로 가리키는 인덱스예요.

내부 반복문은 문자열 b를 처음부터 끝까지 확인해요.
j는 문자열 b의 각 문자를 순서대로 가리키는 인덱스죠.

a의 i번째 문자와 b의 j번째 문자가 같은지 확인해요.
만약 같다면 그 문자를 출력합니다.



문자열 a와 b를 비교해서 두 문자열에 공통으로 있는 문자를 찾아서
화면에 출력합니다.
예를 들어, 'q', 'w', 'e', 'r'과 'q', 'w', 't', 'e', 't', 'y'를 비교하면,
공통 문자 'q', 'w', 'e'가 있고, 이 문자들이 출력되는 거죠.

그래서 정답은 qwe가 돼요.

출력!

[기출 예제] 22년 2회 실기 (난이도: ★★☆)

▶ 정답 10

다음 C언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오.

문자열의 길이를 계산하고, 두 문자열 길이의 합을 출력하는 문제에요.
전혀 어렵지 않아요!

```c
#include <stdio.h>

int len(char*p);

int main() {
    char* p1 = "2022";
    char* p2 = "202207";

    int a = len(p1);
    int b = len(p2);

    printf("%d", a + b);
}

int len(char*p){
    int r = 0;
    while(*p != '\0'){
        p++;
        r++;
    }
    return r;
}
```

len 함수를 위한 함수 선언이에요.
len 함수는 문자열의 길이를 반환하는 함수로, 문자열은 char* 포인터로 전달돼요.

두 개의 문자열 p1과 p2를 선언하고 할당합니다.

함수를 호출하여 각 문자열의 길이를 계산하고, 그 결과를 변수 a와 b에 저장합니다.

a = 4, b = 6으로 10이 출력돼요.

문자열의 길이를 저장할 변수 r을 0으로 할당합니다.

*p가 널 문자가 아닐 때까지 반복합니다. *p는 포인터 p가 가리키는 문자를 나타냅니다.

포인터 p를 다음 문자로 이동시키고,

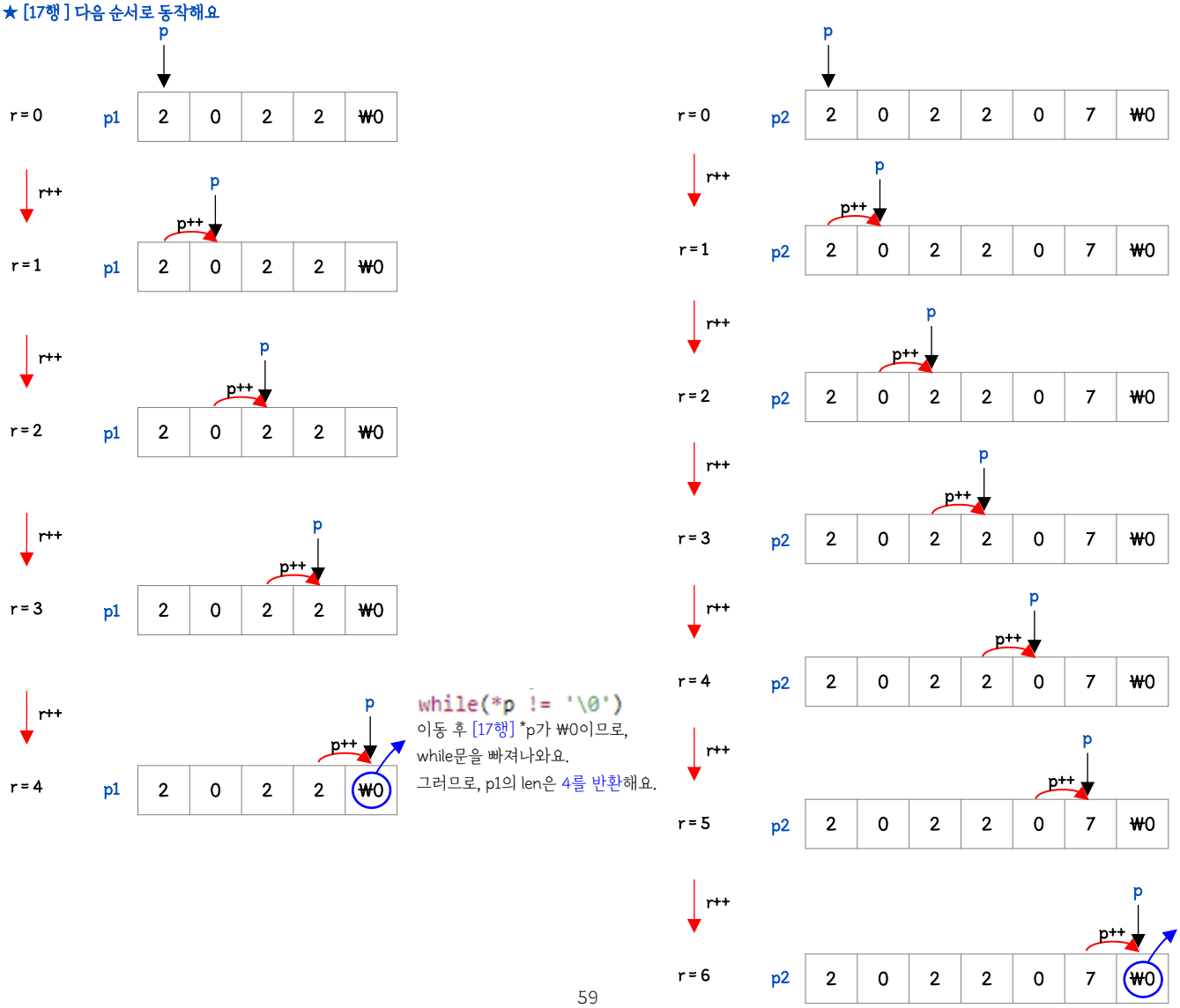
길이 카운터 r을 1 증가시킵니다.

계산된 문자열 길이 r을 반환합니다.

★ key point

후위 연산자 ++: 변수++ 는 변수를 먼저 반환하고, 변수에 변수 + 1를 할당을 뜻해요.
배열의 주소(포인터 p)에 +1을 하면, 다음 문자를 가리킨다는 개념을 알고 있으면 돼요.
위에서 *(a+1)의 출력하라 와 같은 문제처럼 주소 이동에 대한 개념을 묻는 문제에요.
문제 형태는 다르지만, 문제에서 묻고자하는 본질은 같아요

★ [17행] 다음 순서로 동작해요



while(*p != '\0')
이동 후 [17행] *p가 ₩0이므로,
while문을 빠져나와요.
그러므로, p1의 len은 4를 반환해요.

[기출 예제] 23년 2회 실기 (난이도: ★★☆)

다음 C언어로 구현된 프로그램을 분석하여 "홍길동", "김철수", "박영희"를 차례로 입력했을 때 그 실행 결과를 쓰시오.

▶ 정답
박영희
박영희
박영희

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  char n[30];
5  char* getNames() {
6      printf("입력하세요 : ");
7      gets(n);
8      return n;
9  }
10
11 int main() {
12     char * test1;
13     char * test2;
14     char * test3;
15
16     test1 = getNames();
17     test2 = getNames();
18     test3 = getNames();
19
20     printf("%s\n",test1);
21     printf("%s\n",test2);
22     printf(%s,test3);
23 }
```

여기서는 'n'이라는 이름의 문자 배열을 만들고 있어요.
이 배열은 최대 30개의 문자를 저장할 수 있어요.
사용자로부터 이름을 입력받아 그 이름을 다시 돌려주는 역할을 해요.
만약, test1,test2,test3이 같은 메모리 주소가 아니라 별도로 메모리를 할당하고 싶다면,
[4행]을 지우고, [6행] 아래에 char* n = (char*)malloc(30 * sizeof(char)); 로 대체하면 돼요.

사용자의 입력을 받아 'n'이라는 배열에 저장하는 일을 하고 있어요.
'n' 배열에 저장된 값을 함수를 호출한 곳으로 돌려주는 역할을 해요.

'test1', 'test2', 'test3'이라는 이름의 포인터 변수를 선언하고 있어요.
포인터는 메모리 주소를 저장하는 변수로,
여기서는 문자열의 주소를 저장하게 될 거예요.

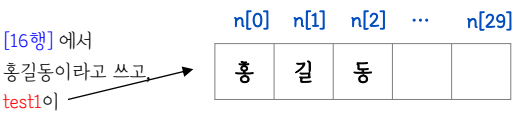
'getNames' 함수를 호출해 사용자로부터 입력받은 이름을
'test1', 'test2', 'test3' 이 가리키게 해요.
이 변수들은 전부 **같은 포인터 (같은 메모리 주소)를 바라보고 있어요.**
이 말은 같은 메모리 주소를 전부 바라 보고 있다면, 마지막으로 그 메모리 주소에
수정한 내용으로 test1, test2, test3도 보이게 될거에요.

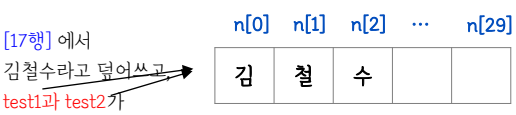
'test2'에 저장된 이름을 화면에 출력해요.
'test3'에 저장된 이름을 화면에 출력해요.

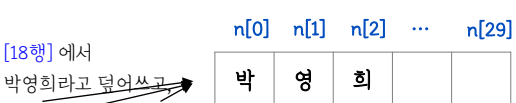
처음에 메모리 x번 위치에 홍길동이라고 저장한 뒤,
x번 위치라는 정보를 test1에 저장 했어요.
두번째로 메모리 (위와 동일한) x번 위치에 김철수 라고 저장한 뒤,
X번 위치를 test 2에 저장 했어요.
현재까지, test1도 같은 메모리 주소를 바라보고 있으니
test1 값을 출력해도. 김철수라고 보일거에요.

마지막으로 동일한 x번 위치에, 박영희라고 저장한 뒤, test3에 포인터를 저장했는데,
test1, test2, test3 전부 같은 메모리 주소를 바라보고, 마지막으로 그 메모리에는
박영희가 저장되어있으니, 전부 박영희로 출력돼요!

★ 포인터
메모리 주소를 다루기 위해 포인터라는 개념을 사용해요.
포인터는 변수가 아닌, 변수의 주소를 저장하는 변수에요.
예를 들어, 'char *'는 문자의 주소를 가리키는 포인터라고 할 수 있어요.







[기출 예제] 22년 2회 실기 (난이도: ★★☆)

다음 C언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오.

▶ 정답
22

```c
#include <stdio.h>

int main() {
    int a[4] = {0, 2, 4, 8};
    int b[3] = {};
    int i = 1;
    int sum = 0;
    int *p1;

    for (i; i < 4; i++) {
        p1 = a + i;
        b[i-1] = *p1 - a[i-1];
        sum = sum + b[i-1] + a[i];
    }

    printf("%d", sum);

    return 0;
}
```

a 배열은 {0, 2, 4, 8}으로 할당돼요.
b 배열은 크기가 3인데, 모든 요소가 0으로 할당돼요.

1부터 시작해서 3까지 반복해요. (1,2,3)

a 배열의 i번째 요소를 가리키게 돼요. 즉, a[i]와 같아요.

b 배열의 (i-1)번째 요소에 a 배열의 i번째 요소에서 i-1번째 요소를 뺀 값을 저장해요.
i = 1 -> b[0] = *p1 - a[0] = 2 - 0 = 2
i = 2 -> b[1] = *p1 - a[1] = 4 - 2 = 2
i = 3 -> b[2] = *p1 - a[2] = 8 - 4 = 4
으로, 값이 할당돼요.

sum을 출력해요. 정답은 22에요.

sum에는 이전 자기 자신 sum, b[i-1]과 a[i]의 합을 더해요.
i = 1 -> sum = 0 + b[0] + a[1] = 0 + 2 + 2 = 4
i = 2 -> sum = 4 + b[1] + a[2] = 4 + 2 + 4 = 10
i = 3 -> sum = 10 + b[2] + a[3] = 10 + 4 + 8 = 22
으로, 값이 할당돼요.

a[0]	a[1]	a[2]	a[3]
0	2	4	8

b 배열의 (i-1)번째 요소에 a 배열의 i번째 요소에서 i-1번째 요소를 뺀 값을 저장해요.
i = 1 ▶ b[0] = *p1 - a[0] = a[1] - a[0] = 2 - 0 = 2 (← *p1 = a[i] 이니까요!)
i = 2 ▶ b[1] = *p1 - a[1] = a[2] - a[1] = 4 - 2 = 2
i = 3 ▶ b[2] = *p1 - a[2] = a[3] - a[2] = 8 - 4 = 4
으로, 값이 할당돼요.

b[0]	b[1]	b[2]
2	2	4

sum에는 이전 자기 자신 sum, b[i-1]과 a[i]의 합을 더해요.
i = 1 ▶ sum = 0 + b[0] + a[1] = 0 + 2 + 2 = 4
i = 2 ▶ sum = '직전 sum 값' + b[1] + a[2] = 4 + 2 + 4 = 10
i = 3 ▶ sum = '직전 sum 값' + b[2] + a[3] = 10 + 4 + 8 = **22 (최종 결과값)**
으로, 값이 할당돼요.

[기출 예제] 24년 1회 실기 (난이도: ★★★)

▶ 정답
Nd sc 1

다음 C언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오.

```c
#include<stdio.h>
#include<ctype.h>

int main(){
    char*p = "It is 8";
    char result[100];
    int i;

    for(i=0; p[i]!='\0'; i++){
        if(isupper(p[i]))
            result[i] = (p[i]-'A'+5)% 26 + 'A';
        else if(islower(p[i]))
            result[i] = (p[i]-'a'+10)% 26 + 'a';
        else if(isdigit(p[i]))
            result[i] = (p[i]-'0'+3)% 10 + '0';
        else if(!(isupper(p[i]) || islower(p[i]) || isdigit(p[i])))
            result[i] = p[i];
    }

    result[i] = '\0';
    printf("%s\n",result);

    return 0;
}
```

문자열 p의 각 문자를 순회하면서 다음 조건에 따라 변환합니다.

대문자인 경우,(isupper 함수는 문자가 대문자인지 여부를 확인)
p[i] - 'A' 현재 문자의 ASCII 값을 기준으로 'A'의 ASCII 값을 빼서 'A'를 0으로, 'B'를 1로, ..., 'Z'를 25로 만들어요.
거기서, +5을 해 5만큼 이동시켜요.
모듈러 연산자 %를 이용해 0~25 범위를 벗어나지 않도록 해요.
모듈러 연산자는 나머지를 뜻해요. 즉, 26로 나눈 나머지 값이에요.
+ 'A' 을 더해 A로 부터 몇만큼 떨어져 있는 수를 구해요.

계산을 해보면, I는 ASCII 값으로 73 A는 65입니다.
(73 – 65 + 5) % 26 + 65 = 78이 되고, 78은 ASCII 값으로 N이 돼요.

이렇게 하나하나 로직을 풀어보면 어렵지만,
결과적으로는 현재 문자열의 5번째 뒤에 있는 문자를 구하는 방법이에요.
I -> J(1) -> K(2) -> L(3) -> M(4) -> N(5)

소문자인 경우,(islower 함수는 문자가 소문자인지 여부를 확인)
대문자와 동일하지만 + 10이므로, 현재 문자 기준으로 10칸 떨어진 소문자를 반환해요.
t -> u(1) -> v(2) -> w(3) -> x(4) -> y(5) -> z(6) -> a(7) -> b(8) -> c(9) -> d(10)
i -> j(1) -> k(2) -> l(3) -> m(4) -> n(5) -> o(6) -> p(7) -> q(8) -> r(9) -> s(10)
s -> t(1) -> u(2) -> v(3) -> w(4) -> x(5) -> y(6) -> z(7) -> a(8) -> b(9) -> c(10)

즉,
t는 d로
i는 s로
S는 c로 변환돼요.

숫자인 경우,(isdigit 함수는 문자가 숫자인지 여부를 확인)
소문자, 대문자와 동일하지만 + 3이므로, 현재 숫자 기준으로 3칸 떨어진 숫자를 반환해요.

즉, 8 + 3으로 11이 되지만, 모듈러 연산자로 나머지인 1이 반환돼요.

그 외의 경우에는 문자 그대로 반환해요.

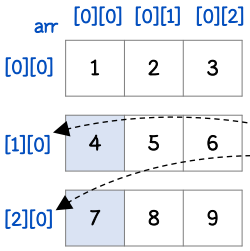
모든 문자열에 대해서 변환을 거치면,
I -> N
t -> d
공백 -> 공백
i -> s
s -> c
공백 -> 공백
8 -> 1

즉, Nd sc 1로 변환되어 출력돼요.
ASCII 코드와 **반복되는 변환 규칙**을 파악해야 풀 수 있는 문제였는데요.
이런 류의 문제를 만나면 로직의 유사성을 눈치채고, 어떤 구조로 변환시키는 지 파악하는 게 중요해요.

만약, 실제 시험장에서 로직 해석이 어렵다면 일단 넘기고 다른 문제를 다 풀고 돌아와서 천천히 파악 후 풀어보는 걸 추천드려요.

[기출 예제] 24년 2회 실기 (난이도: ★★☆)

다음은 C언어에 대한 문제이다. 아래 코드를 확인하여 알맞는 출력값을 작성하시오.

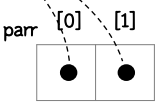


▶ 정답
21

```c
#include <stdio.h>

int main() {
    int arr[3][3] = {1, 2, 3, 4, 5, 6, 7, 8, 9};
    int* parr[2] = {arr[1], arr[2]};
    printf("%d", parr[1][1] + *(parr[1]+2) + **parr);

    return 0;
}
```

3행 3열의 2차원 배열 arr을 선언하고 값을 초기화해요.

크기가 2인 포인터 배열 parr은 선언하고 arr 두번째 행, arr의 3번째 행의 주소를 저장해요.
즉, arr[1]은 4, 5, 6의 시작 주소, arr[2]는 7,8,9의 시작 주소를 저장해요.



arr[1][0], arr[2][0]의 **시작 주소를 저장**

parr[1] = arr[2] 이고,
arr[2]에서 2번째(arr[2][1])인
8을 가져와요.

parr[1] = arr[2]에서
두 칸 이동 = arr[2][2]
9를 가져와요.

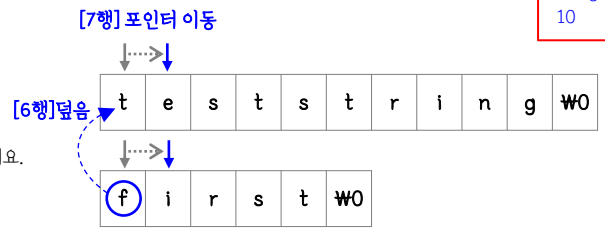
*parr = parr[0]주소 = arr[1] = 4, 5, 6
**parr = arr[1][0] = 4가 돼요.

즉, 8+9+4 = 21이 출력돼요.

[기출 예제] 24년 2회 실기 (난이도: ★★☆)

다음은 C언어에 대한 문제이다. 아래 코드를 확인하여 알맞는 출력값을 작성하시오.

▶ 정답
10




```c
#include <stdio.h>
#include <string.h>

void sumFn(char* d, char* s) {
    while (*s) {
        *d = *s;
        d++;
        s++;
    }
    *d = '\0';
}

int main() {
    char* str1 = "first";
    char str2[50] = "teststring";
    int result=0;
    sumFn(str2, str1);

    for (int i = 0; str2[i] != '\0'; i++) {
        result += i;
    }
    printf("%d", result);

    return 0;
}
```

두 개의 문자열 포인터를 받아요.
"s가 가리키는 곳에 글자가 있는 동안 계속해라"라는 뜻이에요.
s가 가리키는 글자를 d가 가리키는 곳에 할당해요.
d를 다음 위치로 옮겨요.
s를 다음 위치로 옮겨요.
복사가 끝난 후, 문자열의 끝을 나타내는 null 문자를 추가해요.
즉, s를 받아 d에 옮기는 작업을 하는 함수에요.

두 개의 문자열을 만들어요.
str1은 "first"라는 고정 문자열을, str2는 "teststring"이라는 내용으로 시작하는 50자리 배열이에요.
str2에 str1의 값을 덮어써요.

str2의 각 문자에 대해 반복하면서 result에 인덱스 값을 더해요.
문자열의 끝(₩0)에 도달할 때까지 계속해요.

결과적으로 "first"의 길이인 5에 대해 0+1+2+3+4의 합인 10을 출력해요.

[기출 예제] 24년 3회 실기 (난이도: ★★☆)

다음은 C언어에 대한 문제이다. 아래 코드를 확인하여 알맞는 출력값을 작성하시오.

★ key point
포인터와 **2중 포인터**의 동작 원리를 아는지 체크하는 문제에요.
배열의 주소와 포인터 간의 관계를 잘 아는 게 중요해요.

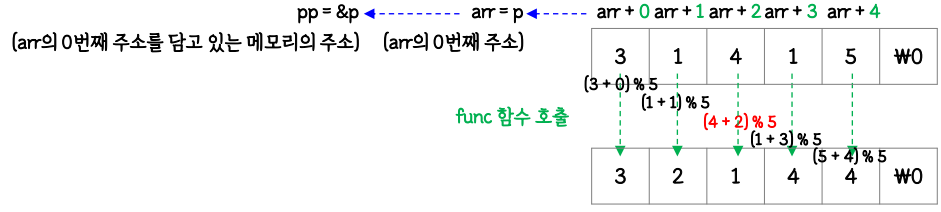
▶ 정답
1

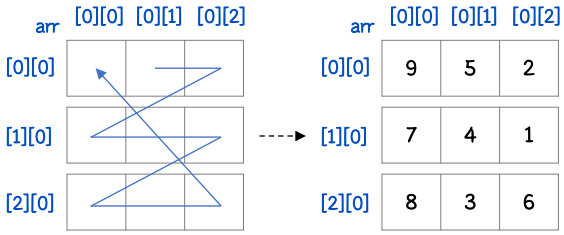
```c
#include <stdio.h>

void func(int** arr, int size) {
  for (int i = 0; i < size; i++) {
    *(*arr + i) = (*(*arr + i) + i) % size;
  }
}

int main() {
  int arr[] = {3, 1, 4, 1, 5};
  int* p = arr;
  int** pp = &p;
  int num = 6;

  func(pp, 5);
  num = arr[2];
  printf("%d", num);

  return 0;
}
```

이중 포인터(arr)과 배열 크기(size)를 매개변수로 받아요.
배열의 크기만큼 반복하면서 요소에 접근해요.
어려우니 천천히 설명해볼게요.
매개변수 arr은 아래 [12행]의 pp에요. 즉, *arr = *pp = *(&p)로 볼 수 있어요.
즉, arr의 0번째 주소가 돼요. (정확하게는 0번째 주소의 메모리 주소의 값 이니 0번째 주소인거죠)
거기에 + i는 arr의 i번째 주소를 가리키는 거구요. (*arr + i)
*(*arr + i) 은 그 주소의 값을 가리켜요.
즉, *(*arr + i) = (*(*arr + i) + i) % size 는 arr의 i 번째 주소의 값을 바꾸는 데, i 번째 주소의 값 + i를 size로 나눈 나머지 값으로 바꾸겠다는 의미에요.

그냥 num 변수 선언 과정이에요.

arr[2] 에는 1이 할당되게 되고,
출력은 1이 돼요.





[기출 예제] 25년 1회 실기 (난이도: ★★★)

다음은 C언어에 대한 문제이다. 아래 코드를 확인하여 알맞는 출력값을 작성하시오.

▶ 정답
13

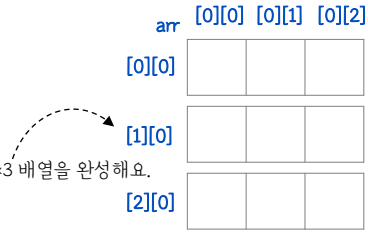
```c
#include <stdio.h>
#include <stdlib.h>

void set(int** arr, int* data, int rows, int cols) {
    for (int i = 0; i < rows * cols; ++i) {
        arr[((i + 1) / rows) % rows][(i + 1) % cols] = data[i];
    }
}

int main() {
    int rows = 3, cols = 3, sum = 0;
    int data[] = {5, 2, 7, 4, 1, 8, 3, 6, 9};
    int** arr;
    arr = (int**) malloc(sizeof(int*) * rows);
    for (int i = 0; i < cols; i++) {
        arr[i] = (int*) malloc(sizeof(int) * cols);
    }

    set(arr, data, rows, cols);

    for (int i = 0; i < rows * cols; i++) {
        sum += arr[i / rows][i % cols] * (i % 2 == 0 ? 1 : -1);
    }

    for(int i=0; i<rows; i++) {
        free(arr[i]);
    }
    free(arr);

    printf("%d", sum);
}
```

→ 아래 main 흐름을 먼저 읽어주세요.

→ for 문은 전체 원소 개수(rows * cols)만큼 반복해요. (3×3 = 9번 반복)

→ ((i+1)/row) % row는 ((i+1)/3) % 3 이고, i+1을 나눈 몫을 3으로 나눴을 때 나머지를 뜻해요.
즉, 첫번째 행부터 채우겠다는 의미에요. 0 -> 0 -> 1 -> 1 -> 2 -> 2 -> 2 -> 0

(i+1)%cols는 i+1을 3으로 나눈 나머지를 뜻해요. 즉, 열 1,2,0 이 반복돼요. (i를 넣어보면 알 수 있어요)



→ 3행 3열의 2차월 배열 arr을 선언하고 값을 초기화해요.

→ 3×3짜리 배열에 채울 9개의 숫자가 배열에 할당되어 있어요.

→ 이중 포인터를 써서 **2차원 배열**을 가리킬 준비를 해요.

→ 행의 개수(rows)만큼 포인터를 담을 포인터 배열을 생성해요.

→ 각 행마다 cols만큼 정수 저장 공간을 따로 할당해, 3×3 배열을 완성해요.

→ arr 이차원 배열이 data 값으로 채워져요.

→ 0부터 8(9미만)까지 순회해요.

→ i / rows는 몇 번째 행(row)인지, i % cols는 몇 번째 열(column)인지 정해요.
(i % 2 == 0 ? 1 : -1) 은 i가 짝수일 때는 곱하기 1을 하여 더하고 홀수 일 때는 곱하기 -1을 하여 sum의 값을 빼요.
행은 i / row = i / 3 이고, 0 -> 0 -> 0 -> 1 -> 1 -> 1 -> 2 -> 2 -> 2의 순서를 가져요.
열은 i % cols = i % 3 이고, 1 -> 2 -> 3 -> 1 -> 2 -> 3 -> 1 -> 2 -> 3의 순서를 가져요.
즉, 채워진 순서와 동일하게 진행되며 i가 짝수이면 더하고 홀수이면 빼면 돼요.
9 – 5 + 2 – 7 + 4 – 1 + 8 – 3 + 6 = 13이 sum에 최종 할당돼요.

→ free 는 동적할당 된 메모리를 해제해요.

→ sum은 13이기에 13이 출력돼요.

★ malloc

동적 메모리 할당 함수입니다.
자바에서는 List를 통해 동적으로 쉽게 데이터를 삽입/삭제 할 수 있지만 C에서는 조금 더 복잡하게 size 바이트만큼 연속된 메모리 공간을 힙 영역에서 할당해요.
cols의 크기인 배열을 프로그램이 동작하는 시점에 읽고 생성한다고 생각하면 돼요. 단순하게 malloc를 통해 이차원 배열을 생성했다 라고 생각하면 됩니다.

★ key point

어려운 문제로 이중 포인터, malloc 함수를 알아야 풀 수 있는 문제에요.
배열 할당의 규칙도 중구난방이고, 복잡해서 실수하기 너무 좋은 문제요. 이럴 때는 값을 미리 넣지 말고 **값 할당의 순서를 미리 그려보고** 채워 넣으면 실수를 줄일 수 있어요.

[문제 유형] 정적 변수

★ 정적변수

1. 메모리에 **한 번만 할당**돼요.
   - 정적 변수는 프로그램이 실행될 때 메모리에 할당되고, 프로그램이 종료될 때까지 그 값을 유지해요.
   - 그래서 함수가 여러 번 호출되더라도 새로 초기화되지 않아요.
2. **초기화는 한 번만 이루어져요.**
   - 정적 변수를 선언할 때 초기값을 주면, 프로그램 실행 시 한 번만 초기화돼요.
   - 이후에는 이전에 저장된 값을 그대로 사용해요.

[기출 예제] 24년 3회 실기 (난이도: ★★☆)

▶ 정답 20

다음은 C언어에 대한 문제이다. 아래 코드를 확인하여 알맞는 출력값을 작성하시오.

```c
#include <stdio.h>

int func() {
  static int x = 0;
  x += 2;
  return x;
}

int main() {
  int x = 1;
  int sum = 0;
  for (int i = 0; i < 4; i++) {
    x++;
    sum += func();
  }
  printf("%d", sum);

  return 0;
}
```

- `static int x = 0;` → 정적 변수는 함수가 끝나도 메모리에 남아 값이 유지돼요. 처음에 x는 0으로 초기화돼요.
  함수가 호출될 때마다 x += 2;로 값이 2씩 증가해요.
  최종적으로 x의 값을 반환(return)해요.
- `int x = 1;` → 이 x는 메인 함수 안에서만 사용되는 지역 변수이고, func 함수의 x와는 다른 거예요.
- `int sum = 0;` → 최종적으로 계산된 값을 저장하는 용도로 사용돼요.
- `for (int i = 0; i < 4; i++) {` → 0부터 시작해서 4번 반복해요.
- `x++;` → 메인 함수의 지역 변수 x를 1 증가시켜요 (헷갈리게 하기 위한 변수 - 지역 변수 x는 사용되는 곳이 없어요.)
- `sum += func();` → func 함수를 호출하고, 반환된 값을 sum에 더해요.

func 내부의 정적 변수 x는 호출될 때마다 2씩 증가하니까, 반환값이 계속 바뀌어요:

- 첫 번째 호출: 2 반환
- 두 번째 호출: 4 반환
- 세 번째 호출: 6 반환
- 네 번째 호출: 8 반환

결과적으로 sum은 2 + 4 + 6 + 8 = 20이 돼요.

★ key point

정적 변수(static)의 특성과 지역 변수와의 차이, 그리고 **변수의 스코프(scope)**를 이해하고 있는지 확인하는 문제입니다. 특히, 함수 내부의 정적 변수가 호출될 때 어떻게 값이 유지되는지를 알아보는 게 목적이에요.

[문제 유형] 구조체

★ 구조체

구조체란, 여러 가지 다른 종류의 변수를 하나로 묶어 새로운 형태의 '데이터 뭉치'를 만드는 것이에요.
마치 일상에서 서로 다른 물건들을 하나의 상자에 담아 정리하는 것처럼요!

예를 들어, 친구 한 명을 생각해볼게요. 이 친구에 대해 이름, 나이, 전화번호 등 다양한 정보가 있잖아요?
C언어의 구조체를 사용하면 이런 정보들을 한데 모아 '친구'라는 하나의 구조체로 만들 수 있어요.

```
struct Friend {
    char name[50];
    int age;
    char phone[15];
};
```

구조체 생성
struct Friend myFriend;

직접 접근
myFriend.name, myFriend.age, myFriend.phone처럼 점(.)을 사용해 해당 친구의 정보에 바로 접근할 수 있어요.

구조체 배열로 생성하기
struct Friend myFriends[5];

배열 접근
myFriends[0].name 이렇게 배열의 인덱스를 사용하여 접근하면 돼요. 여기서 0은 배열의 첫 번째 친구를 의미합니다.

구조체 포인터로 생성하기
struct Friend *ptrFriend;
ptrFriend = &myFriend;

포인터 접근
ptrFriend->name, ptrFriend->age, ptrFriend->phone처럼 화살표(->)를 사용해 포인터가 가리키는 구조체의 정보에 접근하면 돼요.

[기출 예제] 22년 2회 실기 (난이도: ★★☆)

▶ 정답 2

다음 C언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오.

```
1  #include <stdio.h>
2
3  struct A {
4    int n;
5    int g;
6  };
7
8  int main() {
9    struct A a[2];
10   for(int i = 0; i < 2; i++) {
11     a[i].n = i;
12     a[i].g = i+1;
13   }
14   printf("%d", a[0].n + a[1].g);
15 }
```

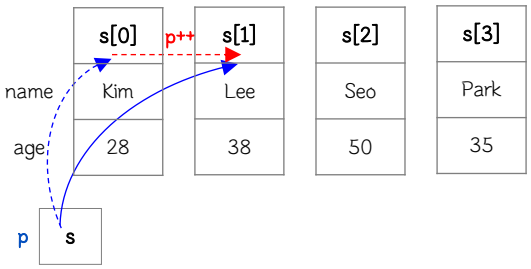
- 3~6행: struct A라는 새로운 데이터 타입을 정의합니다. 이 타입에는 n과 g라는 두 개의 int 타입의 필드(멤버 변수)가 있어요.
- 9행: struct A 타입의 객체 두 개를 담을 수 있는 배열 a를 선언합니다.
- 10행: 0부터 시작하여 2 미만인 경우에 대해 반복하는 루프입니다. 배열 a의 각 원소에 접근하여 값을 설정하는 데 사용돼요.
- 11행: 배열 a의 i번째 원소의 n 필드에 i 값을 할당합니다.
- 12행: 배열 a의 i번째 원소의 g 필드에 i+1 값을 할당합니다.
- 14행: a[0].n의 값과 a[1].g의 값을 더한 결과를 출력합니다.

루프가 두 번 반복되므로,
a[0].n에는 0이, a[0].g에는 1이 저장되고,
a[1].n에는 1이, a[1].g에는 2가 저장됩니다.
따라서, 마지막에 출력하는 a[0].n + a[1].g는 0 + 2이므로 결과는 2가 됩니다.

	a[0]	a[1]
n	0	1
g	1	2

[기출 예제] 21년 1회 실기 (난이도: ★★☆)

다음 C언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오.
(단, 출력문의 출력 서식을 준수하시오.)



▶ 정답
Lee
38

```c
#include <stdio.h>

struct good {
    char name[10];
    int age;
};

void main() {
    struct good s[] = {{"Kim", 28},{"Lee", 38},{"Seo", 50},{"Park", 35}};


    struct good *p;
    p = s;
    p++;

    printf("%s\n", p-> name);
    printf("%d\n", p-> age);
}
```

good이라는 이름의 구조체가 정의되어 있어요.
이 구조체에는 name과 age라는 두 필드가 있어요.

s라는 good 구조체 배열이 할당되어 있어요. 배열은 "Kim", 28, "Lee", 38, "Seo", 50, "Park", 35 이렇게 4명의 정보를 포함하고 있어요.

p라는 포인터를 선언하고,
good 구조체 배열 s의 첫 번째 원소를 가리키게 설정해요.
p++를 통해 포인터를 다음 구조체 원소로 이동시켜요. p = s이니, s+1과 동일하게, 두번째 요소를 가리키도록 움직여요.
즉, p는 이제 배열의 두 번째 원소, 즉 "Lee"와 38을 가리키고 있어요.

★ key point
구조체와 포인터가 둘 다 쓰였어요.
구조체 배열의 첫 번째 주소를 p에 할당했어요.
주소를 +1([14행]) 시키면 다음 요소를 반환한다는 개념은 동일해요.

[기출 예제] 23년 3회 실기 (난이도: ★★☆)

C언어에서 구조체의 멤버에 접근하기 위해 괄호안의 기호를 작성하시오.

출력결과

Lee
41

▶ 정답 ->

```c
#include <stdio.h>

struct insa {
    char name[10];
    int age;
    struct insa* impl_a;
    struct insa* impl_b;
};

int main() {
    struct insa p1 = { "Kim", 28, NULL, NULL };
    struct insa p2 = { "Lee", 36, NULL, NULL };
    struct insa p3 = { "Park", 41, NULL, NULL };

    p1.impl_a = &p2;
    p2.impl_b = &p3;

    printf("%s\n", p1.impl_a( )name);
    printf("%d", p2.impl_b( )age);
}
```

insa라는 구조체를 만들고 있어요. 이름과 나이를 저장할 수 있고, impl_a와 impl_b라는 포인터도 있어요.
포인터는 다른 insa 구조체를 가리키는데 사용돼요.

'Kim', 'Lee', 'Park'이라는 이름을 가진 세 명의 insa 구조체를 만들어요.
각각 나이도 설정하고 있고,
impl_a와 impl_b 포인터는 아직 가리키는 곳이 없어서 NULL로 설정했어요.

p1의 impl_a 포인터가 p2를 가리키게 하고,
p2의 impl_b 포인터가 p3를 가리키게 설정하고 있어요.

p1의 impl_a 포인터가 가리키는 구조체의 이름을 출력하고 (p2의 이름이겠죠?)
p2의 impl_b 포인터가 가리키는 구조체의 나이를 출력해요 (p3의 나이겠죠?).

구조체 포인터의 내부 변수를 접근할 때는 ->를 써요.
기억이 안난다면, 위 페이지의 ★ 구조체에 대한 설명 다시 한번 보시는 걸 추천드려요!

★ key point
구조체의 내부 변수에 접근하는 방법은 일반 변수로 선언했다면, .(점) 구조체 포인터로 선언했다면 ->로 접근이 가능해요.
[6행] [7행]에서 구조체의 주소를 받도록 선언되어 있으니, 바로 정답 -> 를 유추 할 수 있어요.

[기출 예제] 21년 3회 실기 (난이도: ★★☆)

▶ 정답 501

다음 C언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오.
(단, 출력문의 출력 서식을 준수하시오.)

```c
#include <stdio.h>

struct jsu {
  char name[12];
  int os, db, hab, hhab;
};

int main() {
   struct jsu st[3] = {{"데이터1", 95, 88}, {"데이터2", 84, 91}, {"데이터3", 86, 75}};
   struct jsu* p; 
   p = &st[0];

   (p+1)->hab = (p+1)->os + (p+2)->db;
   (p+1)->hhab = (p+1)->hab + p->os + p->db;

   printf("%d\n", (p+1)->hab + (p+1)->hhab);
}
```

st[0]의 주소

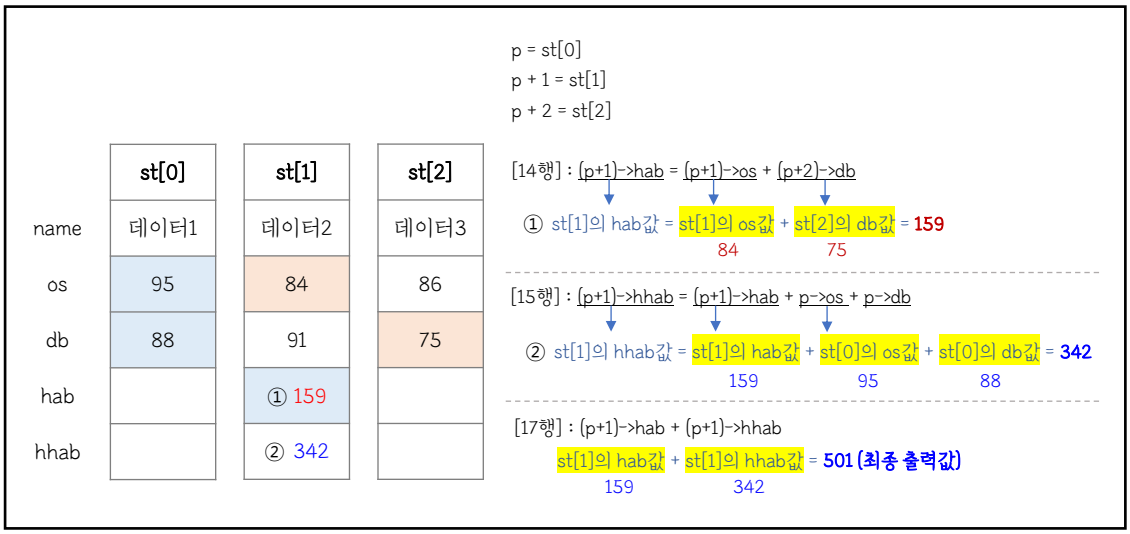
st라는 이름의 jsu 구조체 배열이 3개의 원소로 할당되어 있어요. 각 원소에는 이름과 os, db 값이 들어가 있어요.
hab과 hhab은 할당되지 않아서 기본값은 0이에요.

p; → st 배열의 첫 번째 원소를 가리키고 있어요.

p = &st[0]; → st 배열의 첫 번째 원소를 가리키고 있어요.

[14행] st 배열의 두 번째 원소의 hab 값을 계산해요.
p는 st[0], (p + 1)은 st[1], (p + 2)는 st[2] 를 의미해요.
즉, st[1]의 hab를 st[1]의 os + st[2]의 db으로 할당해요.
이 값은 데이터2의 os 값(84)와 데이터3의 db 값(75)를 더한 159에요.

[15행]은 다음과 같아요.
st[1]->hhab = st[1]->hab + st[0]->os + st[0]->db
데이터2의 hab 값(159)과 데이터1의 os(95)와 db(88) 점수를 모두 더한 값 342에요.
출력은 159 + 342 인 501이 정답 입니다.



[기출 예제] 24년 1회 실기 (난이도: ★★☆)

다음 C언어의 알맞는 출력값을 작성하시오.

▶ 정답
9981 and 2795.10

```c
#include <stdio.h>

typedef struct{
    int accNum;
    double bal;
} BankAcc;

double sim_pov(double base, int year) {
    int i;
    double r = 1.0;

    for (i = 0; i < year; i++) {
        r = r * base;
    }

    return r;
}

void initAcc(BankAcc *acc, int x, double y){
    acc -> accNum = x;
    acc -> bal = y;
}

void update_balance(BankAcc *acc, double *en){
    if (*en > 0 && *en < acc -> bal) {
        acc -> bal = acc -> bal-*en;
    }else{
        acc -> bal = acc -> bal+*en;
    }
}

void apply_interest(BankAcc *acc){
    acc -> bal = acc -> bal * sim_pov((1+0.1),3) ;
}

int main(){
    BankAcc myAcc;
    initAcc(&myAcc, 9981, 2200.0);
    double amount = 100.0;
    update_balance(&myAcc, &amount);
    apply_interest(&myAcc);
    printf("%d and %.2f", myAcc.accNum, myAcc.bal);
    return 0;
}
```

BankAcc라는 이름의 이 구조체는 계좌 번호(accNum)와 잔액(bal)을 가집니다.

base의 year만큼의 거듭제곱 값을 계산하여 반환해요

r은 1.0으로 할당되어 시작해요.

r에 base를 yaer만큼 곱한 뒤 반환해요.

acc 객체와 x, y를 받아
acc.accNum에 x를 할당
acc.bal에 y를 할당해요.

주어진 금액 *en이 양수이고 계좌 잔액보다 작으면
빼고(출금),

그렇지 않으면 더해요. (입금)

acc의 잔액을 1.1(1+0.1)으로 3제곱한 값으로 업데이트해요.

myAcc라는 이름의 BankAcc 구조체 변수를 선언해요.
initAcc 함수를 호출하여 계좌 번호(accNum) 9981과 잔액(bal) 2200으로 초기화해요.
amount 변수를 100으로 초기화해요.
[40행] 함수를 호출하여 100을 출금하여 잔액을 2100으로 만듭니다.
(잔액보다 작은 금액을 넘겼기 때문에 2100으로 차감돼요.)

[41행] 함수를 호출하여 잔액을 업데이트합니다.
결과적으로 잔액은 2100 * 1.331(1.1*1.1*1.1) = 2795.10이 됩니다.

계좌 번호와 잔액을 출력하여 9981 and 2795.10이 정답이 돼요.

★ key point
정말 어려워보이지만,
구조체 개념만 잘 알고 천천히 코드를 해석한다면
크게 어려움 없이 풀 수 있어요.

[기출 예제] 24년 2회 실기 (난이도: ★☆☆)

다음은 C언어의 구조체에 대한 문제이다. 아래 코드를 확인하여 알맞는 출력 값을 작성하시오.

▶ 정답
20

```c
#include <stdio.h>

struct node {
    int n1;
    struct node *n2;
};

int main() {

    struct node a = {10, NULL};
    struct node b = {20, NULL};
    struct node c = {30, NULL};

    struct node *head = &a;
    a.n2 = &b;
    b.n2 = &c;

    printf("%d\n", head->n2->n1);

    return 0;
}
```

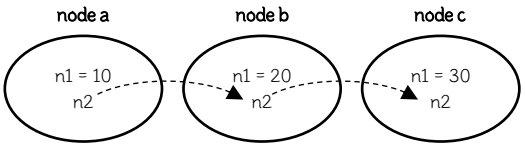
'node'라는 구조체를 정의해요. 이 구조체는 정수 n1과 다른 node를 가리키는 포인터 n2를 가지고 있어요.

세 개의 node 구조체 a, b, c를 만들고 초기화해요.
각각의 n1은 10, 20, 30으로 설정되고, n2는 NULL(아무것도 가리키지 않음)로 설정돼요.

head라는 포인터를 만들고 a의 주소를 저장해요. 이제 head는 a를 가리켜요.

a의 n2가 b를 가리키도록, b의 n2가 c를 가리키도록 설정해요.

head는 a를 가리키고, a의 n2는 b를 가리켜요.
b의 n1은 [11행]에서 20으로 할당했어요.
20이 출력돼요.



▶ 정답
312

[기출 예제] 24년 3회 실기 (난이도: ★★☆)

다음은 C언어의 구조체에 대한 문제이다. 아래 코드를 확인하여 알맞는 출력 값을 작성하시오.

★ key point
구조체와 포인터를 활용한 **연결 리스트의 조작**에 대해서 묻는 문제에요.

구조체 포인터를 사용하는 방법과 이를 이용한 값 변경 로직을 이해하고 있는지 확인해요. (swap 로직과 유사해요)

```c
#include <stdio.h>

struct Node {
  int value;
  struct Node* next;
};

void func(struct Node* node) {
  while (node != NULL && node->next != NULL) {
    int t = node->value;
    node->value = node->next->value;
    node->next->value = t;
    node = node->next->next;
  }
}

int main() {
  struct Node n1 = {1, NULL};
  struct Node n2 = {2, NULL};
  struct Node n3 = {3, NULL};

  n1.next = &n3;
  n3.next = &n2;

  func(&n1);

  struct Node* current = &n1;

  while (current != NULL) {
    printf("%d", current->value);
    current = current->next;
  }

  return 0;
}
```

Node라는 구조체를 정의해요.
- value: 값을 저장하는 변수.
- next: 다음 노드의 주소를 저장하는 포인터

현재 노드가 존재하고, 다음 노드도 존재하는 동안 반복해요.
현재 노드의 값을 임시 변수 t에 저장해요.
현재 노드의 값을 다음 노드의 값으로 바꿔요.
다음 노드의 값을 임시 변수 t에 저장된 값으로 바꿔요.
두 노드씩 이동해요.

노드 n1,n2,n3을 생성하고, 값은 1, 다음 노드는 없음(NULL)으로 설정해요.
n1(1) -> NULL, n2(2) -> NULL, n3(3) -> NULL

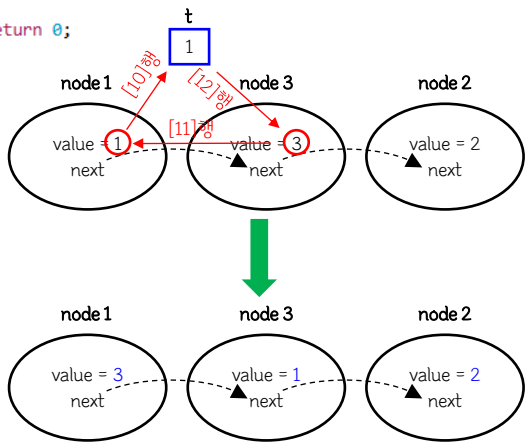
n1(1) -> n3(3) -> n2(2) -> NULL 의 형태로 연결했어요.

n1(3), n3(1), n2(2)의 값이 순서대로 출력되어
312가 출력돼요.

첫 번째 반복:
node는 n1(1)이고, node->next는 n3(3)이에요.
교환: n1.value = 3, n3.value = 1
node는 n2로 이동.
결과: n1(3) -> n3(1) -> n2(2)

두 번째 반복:
node는 n2이고, node->next는 NULL이므로 반복 종료.

최종 리스트:
n1(3) -> n3(1) -> n2(2) -> NULL



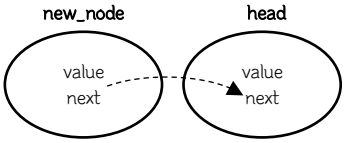
[기출 예제] 25년 1회 실기 (난이도: ★★☆)

▶ 정답
35421

다음은 C언어의 구조체에 대한 문제이다. 아래 코드를 확인하여 알맞는 출력 값을 작성하시오.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  typedef struct Data {
5      int value;
6      struct Data *next;
7  } Data;
8
9  Data* insert(Data* head, int value) {
10     Data* new_node = (Data*)malloc(sizeof(Data));
11     new_node->value = value;
12     new_node->next = head;
13     return new_node;
14 }
15
16 Data* reconnect(Data* head, int value) {
17     if (head == NULL || head->value == value) return head;
18     Data *prev = NULL, *curr = head;
19     while (curr != NULL && curr->value != value) {
20         prev = curr;
21         curr = curr->next;
22     }
23
24     if (curr != NULL && prev != NULL) {
25         prev->next = curr->next;
26         curr->next = head;
27         head = curr;
28     }
29     return head;
30 }
31
32 int main() {
33
34     Data *head = NULL, *curr;
35     for (int i = 1; i <= 5; i++) {
36         head = insert(head, i);
37     }
38     head = reconnect(head, 3);
39     for (curr = head; curr != NULL; curr = curr->next) {
40         printf("%d", curr->value);
41     }
42     return 0;
43 }
```

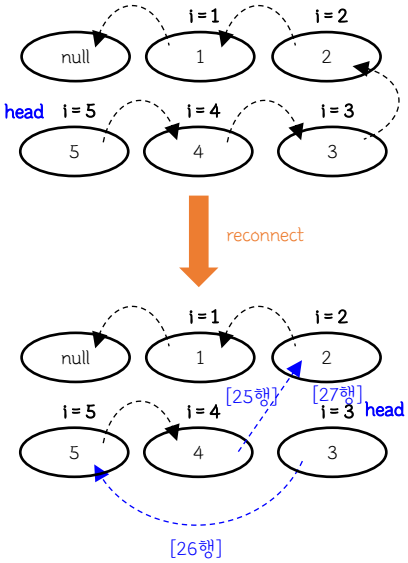
'Data'라는 구조체를 정의해요.
값을 저장할 value와 다음 노드를 가리킬 포인터 next를 변수로 가지고 있어요.



새로운 노드를 만들어요. (malloc는 동적으로 메모리를 size만큼 생성해요. – 자바의 new와 유사)
그 노드의 값을 value로 할당하고, 새로운 노드의 next에 기존 head 노드를 가리키게 합니다.
새로운 노드를 반환해요. (즉, 새로운 노드가 리스트의 맨 앞에 위치하게 돼요.)

[38행]을 확인하고, 어떤 값이 들어오는 지 확인합니다. head = reconnect(head, 3); 코드가 실행됩니다.
만약 리스트가 비었거나(head==NULL), 이미 맨 앞이면 그냥 리턴.
while 반복문이 실행되어 값이 3인 노드(curr)와 그 이전 노드(prev)를 찾습니다.
prev는 i=4인 노드를 가리킵니다.
curr는 i=3인 노드를 가리킵니다.

i = 4 노드의 next 포인터를 i = 3 노드가 가리키던 2번 노드로 변경합니다.
i = 3 노드의 next 포인터를 현재 head인 i = 5 노드로 변경합니다.
리스트의 head를 i = 3 노드로 변경합니다.



head (i=3) 부터 시작하여 null까지 출력합니다.
오른쪽 그림대로 35421 가 출력됩니다.

[문제 유형] 자료 구조

[기출 예제] 23년 2회 실기 (난이도: ★★☆)

▶ 정답 213465

다음 C언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오.

```c
#include <stdio.h>
#define MAX_SIZE 10
int isWhat[MAX_SIZE];
int point= -1;

void into(int num) {
    if (point >= 10) printf("Full");
    else isWhat[++point] = num;
}

int take() {
    if (isEmpty() == 1) printf("Empty");
    else return isWhat[point--];
    return 0;
}

int isEmpty() {
    if (point == -1) return 1;
    return 0;
}

int isFull() {
    if (point == 10) return 1;
    return 0;
}

int main(int argc, char const *argv[]){
    int e;
  ①into(5);②into(2);
    while(!isEmpty()){
      ③printf("%d", take());
      ④into(4);⑤into(1);⑥printf("%d", take());
      ⑦into(3);⑧printf("%d", take());⑨printf("%d", take());
      ⑩into(6);⑪printf("%d", take());⑫printf("%d", take());
    }
    return 0;
}
```

MAX_SIZE 10 스택의 최대 크기를 10으로 정의하는 부분이에요. MAX_SIZE는 스택에 들어갈 수 있는 최대 요소의 수를 나타냅니다.

스택을 저장할 배열 isWhat를 선언합니다. 여기서 최대 10개의 정수를 저장할 수 있어요.

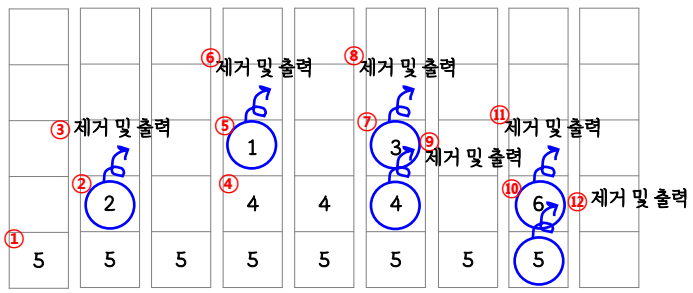
스택의 현재 위치를 나타내는 포인터 point를 선언하고, 초기값으로 -1을 설정합니다. 스택이 비어 있음을 의미해요.

스택에 데이터를 삽입하는 함수예요.
만약, 스택이 가득 차 있으면 "Full"을 출력하고, 아니면 주어진 숫자 num을 스택에 삽입합니다.
point에 +1은 먼저 한 뒤에 반환해요.

스택에서 데이터를 꺼내는 함수입니다.
스택이 비어 있으면 "Empty"를 출력하고, 아니면 스택의 가장 위에 있는 데이터를 꺼냅니다.
스택에서 제거하면서, 그 값을 반환해요.
point을 먼저 반환한 뒤, -1을 더해줍니다.

스택이 비어 있는지 확인하는 함수입니다. 비어 있으면 1을, 아니면 0을 반환합니다.
[31행]에서 스택이 전부 빌 때까지 반복해요.

스택이 비어 있는지 확인하는 함수입니다. 비어 있으면 1을, 아니면 0을 반환합니다.
여기서 사용되지 않아요.



[문제 유형] 2차원 배열

[기출 예제] 22년 3회 실기 (난이도: ★★★)

다음 C언어로 구현된 프로그램을 분석하여 배열 <mines>의 각 칸에 들어갈 값을 쓰시오.

배열 <field>

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

배열 <mines>

★ 전략이 필요해요.
코드 난이도 높고, 문제 풀이 자체에도 시간이 많이 드는 문제에요. 이런 문제가 나오면 일단, 다음 문제 부터 풀고 시간 남으면 돌아와서 문제를 푸는 걸 추천드려요!

▶ 정답
1 1 3 2
3 4 5 3
3 5 6 4
3 5 5 3

```c
#include <stdio.h>

int calculate(int w, int h, int j, int i) {
  if (i >= 0 && i < h && j >= 0 && j < w) return 1;
  return 0;
}

void main() {
    int field[4][4] = {{0,1,0,1},{0,0,0,1},{1,1,1,0},{0,1,1,1}};
    int mines[4][4] = {{0,0,0,0},{0,0,0,0},{0,0,0,0},{0,0,0,0}};

    int w = 4, h = 4;
    for(int y=0; y<h; y++) {
        for(int x=0; x<w; x++) {
        if(field[y][x] == 0) continue;
            for(int i=y-1; i<=y+1; i++) {
                for(int j=x-1; j<=x+1; j++){
                    if(calculate(w,h,j,i) == 1) {
                        mines[i][j] += 1;
                    }
                }
            }
        }
    }
    for(int y=0; y<h; y++){
        for(int x=0; x<w; x++) {
            printf("%d ", mines[y][x]);
        }
        printf("\n");
    }
}
```

주어진 좌표 (i, j)가 배열의 범위 안에 있는지 확인합니다.
w는 배열의 너비, h는 높이를 의미합니다.
좌표가 배열 범위 안에 있으면 1을, 아니면 0을 반환해요.

field 배열은 4x4 크기의 지뢰 필드를 나타냅니다. 1은 지뢰가 있는 칸을, 0은 지뢰가 없는 칸을 의미합니다.

mines 배열은 주변에 몇 개의 지뢰가 있는지를 저장할 배열입니다.
처음에는 모든 값이 0으로 할당 되어 있죠.

w와 h는 게임 필드의 너비와 높이를 나타내는 변수들이에요. 여기서는 둘 다 4로 설정되어 있습니다.

외부 for 루프는 y를 사용해 필드의 행을 반복합니다.

내부 for 루프는 x를 사용해 필드의 열을 반복합니다.

현재 칸에 지뢰가 없으면(field[y][x] == 0), 주변 칸의 지뢰 수를 세는 것을 건너뜁니다.

각 지뢰 칸을 중심으로 나 + 주변의 8개 칸(총 9칸)을 검사합니다.
여기서는 지뢰가 있는 현재 칸의 좌표를 기준으로
상하좌우, 대각선에 있는 칸을 확인합니다.
현재 y 행기준으로 위(y-1) 아래(y+1)까지 검사해요
현재 x 열 기준으로 왼쪽(x-1) 오른쪽(x+1)까지 검사해요

calculate 함수를 호출하여 (i, j) 좌표가 필드의 범위 내에 있는지 확인합니다.
그 결과가 1이면 해당 좌표는 유효합니다.

유효한 좌표라면 mines 배열의 해당 좌표에 1을 더해서
주변 지뢰의 수를 증가시킵니다.

★ 다음과 같은 흐름으로 mines를 채울 수 있어요.

X 표시는 calculate() 결과 유효하지 않음을 뜻해요.
→ 표시는 주위에 지뢰가 있음을 뜻해요.
→ 표시는 주위에 지뢰가 없음을 뜻해요.
○ 표시는 내 위치에 지뢰가 있음을 뜻해요.
○ 표시는 내 위치에 지뢰가 없음을 뜻해요.