🧩 1️⃣ 공통문제 (C) – 문자열 뒤집기

✅ 정답 여부
* 전체적으로 정확하게 구현하셨습니다.
* `for`문을 활용해 뒤에서부터 문자를 차례로 저장한 부분이 잘 구성되었습니다.

⚠️ 개선할 점
* 문자열 종료는 `NULL`이 아니라 `'₩0'` 문자로 처리해야 합니다.
* 또한 `answer` 배열이 동적 메모리로 선언되지 않았고, `malloc`이 빠져 있어서 실행 환경에 따라 **미정의 동작**이 발생할 수 있습니다.

🔧 추천 개선 방향
* `malloc`을 사용해 문자열 크기만큼 동적 메모리를 할당하고, 마지막에 `'₩0'`을 넣어주는 방식으로 안전하게 구현하는 것이 좋습니다.

🧩 2-1. p와 y의 개수 비교 (C++)

✅ 정답 여부
* 문제를 정확하게 해결하셨고, `p`와 `y`의 개수를 올바르게 세신 점이 좋습니다.

⚠️ 개선할 점
* 조건문이 조금 길게 구성되어 있는데, 아래와 같이 간단하게 줄일 수 있습니다.

``cpp return countp == county; ```	cpp return countp == county;

📝 추가 피드백
* 논리적인 판단 결과를 그대로 `return`하는 방식은 코드 가독성과 유지보수 측면에서 더 좋습니다.

🧩 2-2. 괄호 짝 확인 (C#)

✅ 정답 여부
* 대부분의 테스트케이스를 통과할 수 있을 정도로 정확하게 구현하셨습니다.

⚠️ 개선할 점
* 조건문이 복잡하게 구성되어 있어, 코드 흐름을 파악하기 어렵습니다.
* 괄호 짝을 확인할 때는 일반적으로 **스택** 또는 **누적 카운터 방식**을 사용합니다.

📝 추천 개선 방향
* `start`, `end`와 같은 이중 변수보다는 **열린 괄호 수를 카운트**하고,
  닫는 괄호가 더 많아지는 경우에만 `false`를 반환하는 방식으로 처리하면 간결하면서도 안정적인 구현이 가능합니다.

🧩 3-1. 문자열이 숫자로만 구성되었는지 확인 (C++)

✅ 정답 여부
* 조건 판단과 `isdigit()` 함수 사용이 적절하게 이루어졌습니다.

⚠ 개선할 점
* `answer = false;`가 여러 번 반복되는 구조입니다.
* 아래와 같이 조건 충족 시 바로 `return false`하는 구조로 바꾸는 것이 더 간결하고 실용적입니다.

``cpp for (char c : s) { if (!isdigit(c)) return false; } return true; ```	cpp for (char c : s) { if (!isdigit(c)) return false; } return true;

🧩 3-2. JadenCase 문자열 만들기 (C++)

✅ 정답 여부
* 문자열을 단어별로 인식하고 대소문자를 적절히 처리한 점이 좋습니다.

⚠ 개선할 점
* 입력 문자열을 직접 수정하기보다는 새로운 문자열을 생성하는 방식(`ostringstream` 등)을 사용하는 것이 더 안전하고 명확합니다.

📝 종합 피드백 요약

문제	핵심 피드백 요약
1번 (C)	문자열 종료는 `'₩0'` 사용, `malloc` 필요
2-1 (C++)	`return countp == county;`로 간결화 가능
2-2 (C#)	괄호 검사는 누적 카운터 또는 스택이 더 효율적
3-1 (C++)	불필요한 조건문 제거, 즉시 `return false` 구조 추천
3-2 (C++)	`s[i]` 직접 수정보다 새 문자열 생성 방식이 안정적

🧠 추가 성장 방향

* **문제 해결만이 아니라, 구현 방식의 이유와 선택 근거를 설명할 수 있도록 훈련**해 보세요.
* 조건문/반복문 등 기본 로직을 더 간결하게 작성하는 습관을 들이면 실무에서도 큰 도움이 됩니다.
* 코드를 리팩토링해보는 경험을 쌓는 것도 좋은 연습이 됩니다.