**작품 설명서**

Troy팀 주제: LLM 기반 피드백 OJ 시스템

팀장: 장경은 팀원: 손승민 이태우 최연호ssss

**1. 개요**

LLM(대규모 언어 모델)을 활용한 피드백 제공 OJ(Online Judge) 시스템은 기존의 자동 채점 시스템을 확장하여, 코드의 정확성 뿐만 아니라 문제 이해 등에 대한 피드백을 제공하는 시스템입니다. 본 시스템은 AI를 활용하여 코드의 논리적 오류를 분석하고, 학생마다 이해 수준을 고려한 차별화된 개선점을 제안하는 기능을 갖추고 있습니다.

**2. 개발 배경 및 목적**

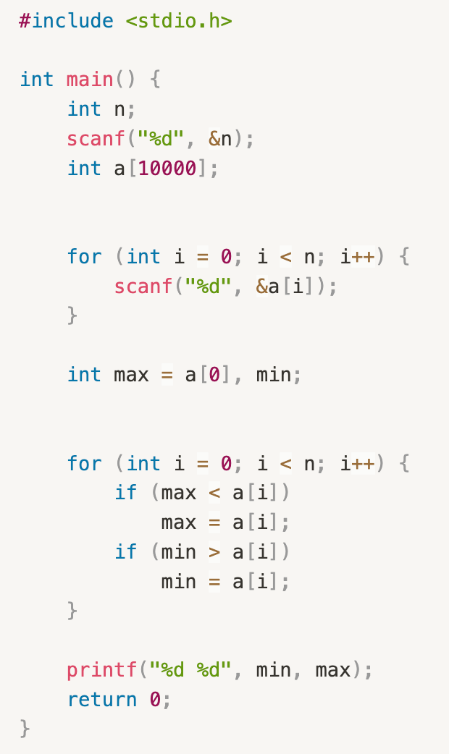
일반적인 OJ 시스템은 정해진 테스트 케이스를 기반으로 코드의 정답 여부만 판단하지만, 사용자의 코드가 왜 틀렸는지 또는 문제 이해 과정에서 어떤 오류가 있었는지에 대해 단계적 피드백을 제공하지 못합니다. 따라서 본 시스템은 LLM을 활용하여 학습자의 문제 이해도와 코드 작성 방식을 고려한 개선된 피드백을 제공함으로써, 각 학생의 문제 해결 능력을 효과적으로 향상시키는 것을 주요 목표로 합니다.

**3. 시나리오**

**3-1. 시나리오1 - 학생 A가 배열 선언을 어려워하는 상황**

반복문에 대한 개념을 배운 학생 A가 문제를 풀고 피드백을 받는 부분을 예시로 들겠습니다.

오늘 배운 반복문을 토대로 아래와 같은 코드를 학생 A가 제출하였습니다



하지만 채점결과를 보니 런타임 에러가 발생해서 피드백을 눌렀습니다

이미지 삽입 중...

피드백 내용은 다음과 같았습니다.

‘’’

***Runtime Error***

***입력 파트 (O)***

* *문제 없습니다!*

***문제풀이 부분 1 (X)***

* *배열의 크기 부분에 문제가 있습니다*
  + *배열 부분에서 메모리 크기할당에서 문제가 발생했습니다.*
  + *이런 경우에는 동적메모리 할당을 사용하는 것이 유용합니다.*

***출력 파트 (O)***

* *문제 없습니다!*

*동적 메모리할당에 대한 부분을 배웠나요? 코드에서 배열을 사용할 때는 먼저 크기를 정한 후 사용하는 것이 중요해요! 현재 코드에서는 배열의 크기가 고정되어 있어 입력 데이터가 유동적일 경우 문제가 생길 수 있답니다. 배열을 사용하기 전에 크기를 결정하고, 필요한 경우 동적 메모리 할당을 활용하면 더 유연한 코드가 될 거예요* [*www.example.com*](http://www.example.com)*에서 메모리 할당에 대한 개념을 확실히 이해해보세요!*

‘’’

위 피드백을 보고 수업시간에 배웠던 메모리 할당의 개념을 잘 이해하지 못했던 학생 A는 사이트에 들어가서 동적 메모리에 대한 개념을 확실히 이해할 수 있었습니다!

**3-2 시나리오2 : 학생 B가 조건문 작성을 어려워하는 상황**

텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

오늘 조건문에 대해 배운 학생 B는 실력 증진을 위해 다음과 같은 문제를 푸는 과정을 예시로 들겠습니다.

이미지 삽입 중...

문제를 푸는 과정에 있어서 코드를 어떻게 짜야할지 몰라서 학생 B는 단계적 피드백을 이용하고자 합니다. 학생 B는 결국, 빈 코드를 제출하였고, 이에 대한 피드백 버튼을 눌러서 내 코드의 피드백 화면을 보았습니다.

텍스트, 스크린샷, 멀티미디어 소프트웨어, 소프트웨어이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

입력 부분에서는 scanf 함수를 사용해 문제에서 요구하는 값을 입력받습니다. 이후 수업 시간에 배운 조건문을 활용하는데, 예를 들어 if (grade >= '범위' && grade <= '범위')와 같은 형태로 조건문을 설정할 수 있습니다. 그런 다음, 출력 부분에서는 printf("출력");을 사용하여 결과를 출력하게 됩니다. 이 과정을 단계적으로 피드백을 통해 이해하고 적용할 수 있도록 안내합니다.

‘’’

***Wrong Answer***

***입력파트 (O)***

*문제없습니다!*

***문제풀이부분1 (X)***

*조건문 부분에 문제가 있습니다.*

***문제풀이부분2 (X)***

*논리 연산자가 잘못 사용되었습니다.*

*두개의 조건을 모두 만족하려면 && 연산자를, 두 조건중의 하나만 만족하면 || 연산자를 사용하여야 하는데 알맞게 쓰였는지 다시 확인해주세요*

***출력파트 (O)***

‘’’

학생 B는 단계적 피드백을 통해서 코드를 통해서 코드의 형식을 입력 부분, 문제 풀이 부분, 출력부분을 따로 분리해서 단계적으로 설명하여 문제를 훨씬 쉽게 접근하여 코드를 구상해 나갈 수 있었고 이 기능을 이용하여 문제를 해결했습니다.

**3-3. 시나리오3 – 학생 C가 반복문과 문제 속 개념 이해가 어려운 상황**

텍스트, 스크린샷, 폰트, 대수학이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.  
두 개의 자연수를 입력받아 최대 공약수와 최소 공배수를 출력해야 하는 문제를 학생 C가 풀어가는 과정을 예시로 들겠습니다.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.문제에서 요구하는 것이 무엇인지 잘 모르겠는 학생 C는 문제 화면에서 마크업 기능을 사용합니다.

위 사진과 같이 마크업을 통해 문제 속에서 이용되는 최대공약수와 최소공배수의 개념을 바로 이해했습니다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

학생 C는 최대공약수와 최소공배수라는 몰랐던 개념을 알게 되었고 해당 개념을 기반으로 위와 같은 코드를 제출했습니다.

학생 C는 제출한 코드에 대해 채점 후 결과를 제출합니다.

학생 C는 결과를 확인했는데 컴파일 에러가 발생해서 피드백 버튼을 눌러 피드백 화면으로 넘어갑니다.

텍스트, 폰트, 스크린샷, 번호이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

LLM모델은 학생 C의 코드를 분석하여 사용자가 어떠한 개념을 어려워하는지와 사용자의 코딩 실력을 판단합니다.

이를 바탕으로 유형별로 정의된 프롬프트와 함께 제출된 코드에 대한 피드백을 작성합니다.

아래는 피드백 예시입니다.

‘’’

***Wrong Answer***

***입력파트 (O)***

*문제없습니다!*

***문제풀이부분1 (X)***

*C변수는 불필요해 보입니다.*

***문제풀이부분2 (X)***

*최대공약수와 최소공배수를 구하는 알고리즘이 잘못되었습니다.*

*최대공약수와 최소공배수의 개념이 헷갈린다면, (*[*www.example.com*](http://www.example.com)*)을 확인하세요!*

*최대공약수를 먼저 구한 후 최소공배수를 구하는 것을 추천드립니다.*

*- 두 수중 작은 값까지 for문을 돌립니다*

*- i가 a와 b를 동시에 나누는지 확인하세요.*

***문제풀이부분3 (X)***

*두 수를 곱한 후 최소공배수로 나누면 최대공약수를 구할 수 있습니다.*

***출력파트 (X)***

*최대공약수와 최소공배수를 한 줄씩 출력하면 됩니다!*

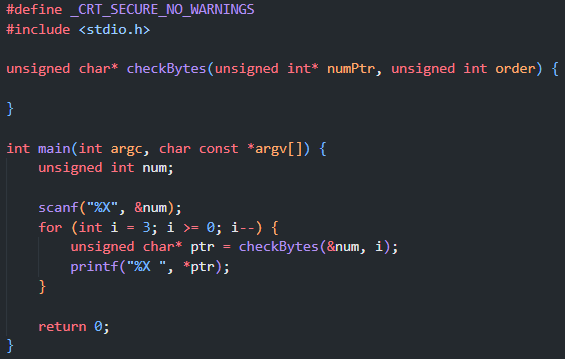
‘’’

**3-4. 시나리오4 – 학생 N이 비트 쉬프트 연산에 대해서 어려워하는 경우**

학생 M은 도전 과제를 풀기 위해 아래의 문제를 풀기로 했다.

*unsigned int형 포인터 numPtr과 순서 order를 인자로 받아서,*

*order번 byte의 주소를 unsigned char형 포인터로 반환하는 함수 checkBytes를 작성하시오.*



하지만 문제 자체를 이해하지 못한 학생 M는 함수 구조만 작성하고 내부 코드를 작성하지 못한 채로 제출하였습니다.

위의 코드를 제출 후 피드백 요청 버튼을 눌러 피드백을 확인하였습니다.

‘’’

***입력 파트 (O)***

*문제 없습니다!*

***문제풀이부분 (X)***

*이 문제에서는 Unsigned int형 정수의 각 바이트를 분리하여 출력해야합니다. 4가지 바이트를 다룰 수 있도록 포인터 연산을 활용해야합니다! unsigned int\*형을 바이트 단위로 접근하려면 unsigned char\*형 포인터로 가리키면 됩니다. 왜냐하면 unsigned int형이 4byte이고, unsigned char형이 1byte이기 때문에 쉬프트 연산을 이용하면 4번 각각 가리킬 수 있기 때문입니다!*

*쉬프트 개념이 헷갈리신다면 이 링크를 참고해보세요! (www.example.com)*

***출력 파트 (O)***

*문제없습니다!*

‘’’

위의 피드백을 참고하여 학생 N은 수정된 코드를 작성하여 OJ 시스템에 제출하였습니다.

**3-5. 시나리오5 – 학생 E가 포인터 연산자를 혼동하고 있는 상황**

학생 E는 이번 주차 이론 시간에 포인터에 대해서 공부하고, 숙제로 아래와 같은 손코딩 문제를 풀려고 합니다.

*입력받은 값을 가리키는 포인터를 만들어서 주소를 출력하는 프로그램을 작성하시오*

텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

학생 E는 int형 변수 num을 가리키는 ptr를 선언하여, 주소를 출력해주는 &연산자를 이용해 ptr의 주소를 출력하였습니다.

그러나 원하는 주소가 아닌, 다른 주소가 출력되었고 정성적 평가에 따라 점수를 받지 못한 학생 E는 OJ 시스템의 피드백 기능을 이용하기로 하였습니다.

‘’’

***Wrong Answer***

***입력파트 (O)***

*문제없습니다!*

***출력부분 (X)***

*ptr가 가리키는 num 변수의 주소값을 확인하기 위해서는 num, &num, ptr, &ptr 각각의 차이점을 숙지해야합니다.*

*printf를 이용하여 각각 출력해보고, 각각 어떤 의미를 지니고 있는지 확인해보세요!*

*포인터 개념이 헷갈리신다면, (*[*www.example.com*](http://www.example.com)*)을 확인해보세요.*

‘’’

학생 E가 위의 피드백을 확인하여 주소값을 모두 확인해본 결과, &num와 ptr가 같은 주소를 가지고 있음을 확인하였습니다.

텍스트, 폰트, 스크린샷이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

따라서 포인터가 지니고 있는 값을 그대로 출력하면 가리키고 있는 변수의 주소값이 출력된다는 것을 이해하였고, 아래와 같이 코드를 수정하여 다시 제출하여 통과하였습니다.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

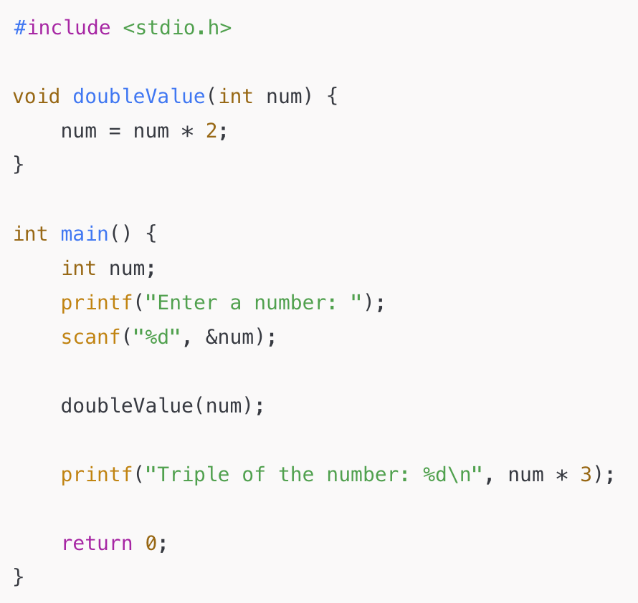
AI가 생성한 콘텐츠는 부정확할 수 있습니다.

**3-6. 시나리오6 – 학생 F가 Call by Reference와 Call by Value를 혼동하고 있는 상황**

학생 F는 포인터의 개념을 배운 후 다음과 같은 문제를 풀려고 합니다.

**int형 변수를 선언하여 사용자에게 입력받은 후 해당 값을 2배로 바꿔주는 함수(반환 형태 : void)를 사용하여 입력받은 수의 3배를 출력하시오**

학생F는 위 문제에 대하여 다음과 같이 코드를 제출하였습니다.



학생 F는 위 코드의 실행결과를 보고 다음과 같이 생각하였습니다.

“왜 내가 함수 안에서 num을 2배로 바꿨는데도 main()에서 출력할 때 바뀌지 않는 거지? 난 값만 넣으면 되는 거 아닌가?”

이해가 잘 되지 않던 학생F는 피드백 버튼을 눌렀습니다.

*‘’’*

***Wrong Answer***

***입력 파트 (O)***

*입력받는 부분에서 문제는 없습니다. 사용자에게 값을 정상적으로 입력받고 있습니다.*

***문제풀이부분1 (X)***

*doubleValue() 함수에서 int num을 값으로 전달하고 있습니다. 이 경우 함수 내부에서 값을 변경하더라도 원본 값은 변경되지 않습니다.**num을 실제값으로 변경하려면 포인터를 사용해야합니다. 경우에 따라 printf를 이용하여 디버깅을 수행하세요! 포인터 개념은 (*[*www.example.com*](http://www.example.com)*)을 확인하세요!*

***출력파트 (X)***

*3배를 출력하려면, 먼저 num을 2배로 바꾼 후 3배로 계산하여 출력해야 합니다.*

*현재는 num 값을 그대로 사용하여 3배로 출력하고 있으므로, 2배 값을 반영한 후 3배를 출력해야 정확한 결과가 나옵니다.*

***‘’’***

학생 F는 피드백을 읽고 원본 변수의 값이 변경되지 않았기 때문에 틀렸다는 점을 깨달았습니다. 원본 변수의 값을 변경하기 위해서는 포인터를 이용해야한다는 사실을 깨닫고 주어진 하이퍼링크에서 개념을 공부했습니다.

그리고 공부한 개념을 토대로 아래와 같이 수정된 코드를 제출하였습니다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

학생 F는 변수의 주소를 넘겨서 포인터로 원본 변수의 값을 변경하였습니다.

수정된 코드를 제출하여 OJ 시스템에서 Accepted를 받았습니다!



**3-7. 시나리오7 – 학생 G가 논리 연산자에 대한 개념이 어려운 상황**

학생 G는 이번 주차 숙제 중 아래와 같은 문제를 풀기로 하였습니다.



학생 G는 영수증 문제를 해결하기 위해서 아래와 같은 코드를 작성하였습니다.

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

하지만 해당 코드를 OJ 시스템에 제출한 결과 Wrong Answer를 받고, 틀린 이유를 도무지 몰라 피드백 버튼을 클릭하였습니다.



그리고 아래와 같은 피드백을 받았습니다.

‘’’

***Wrong Answer***

***입력파트 (O)***

*문제없습니다!*

***문제풀이부분1 (X)***

*현재 코드에서 입력값 검증 조건이 잘못된 논리 구조로 작성되었습니다. for문에서 a와 b를 처리하는 조건이 잘못된 것 같습니다. 다시 한번 확인해보세요!*

***출력파트 (O)***

*문제없습니다!*

‘’’

학생 G는 해당 피드백을 참고하고 마지막에 not연산자를 붙이지 않았음을 깨달았습니다. 따라서 해당 부분을 수정하고 다시 제출하여 OJ 시스템에서 통과하였습니다!

**3-8. 시나리오8 – 학생 H가 변수 범위를 고려하지 못한 경우**

텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

학생 H는 변수값을 변경하는 함수를 작성하라는 문제를 풀기로 하고 위와 같은 코드를 작성하였습니다.

그러나 전역 변수와 지역 변수의 차이점을 몰랐던 H는 지역 변수를 선언해서 변경했습니다. 따라서 변경되지 않은 변수가 출력되었고 왜 틀렸는지 이해할 수 없었습니다.

피드백을 받기 위해 OJ 사이트에 제출 후 피드백 버튼을 클릭하였습니다.

OJ 사이트에서는 아래와 같은 피드백을 받고, 변수 범위를 이해하게 되었습니다.

*‘’’*

***Wrong Answer***

***입력파트 (O)*** *문제없습니다!*

***문제풀이부분1 (X)***  
*전역 변수 x가 존재함에도, changeValue() 함수 내에서 같은 이름의 지역 변수 x를 선언하였습니다. 이로 인해, 지역 변수 x가 전역 변수 x를 덮어쓰게 되어 전역 변수의 값이 변경되지 않습니다.*  
*전역 변수를 변경하려면, 지역 변수를 새로 선언하는 것이 아니라 전역 변수 자체를 수정해야 합니다. 전역 변수를 이용하려면 어떻게 해야할까요?*

***문제풀이부분2 (X)***  
*전역 변수와 지역 변수의 개념이 헷갈린다면, (*[*www.example.com*](http://www.example.com)*)을 참고하세요!*

***출력파트 (O)***

*문제없습니다!*

*‘’’*

학생은 자신이 전역 변수와 지역 변수를 혼동했다는 사실을 깨닫고 지역 변수 선언 부분을 아래와 같이 수정하였습니다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

전역 변수와 지역 변수의 개념을 온전히 이해한 학생 H는 해당 문제를 통과했습니다!



**3-9. 시나리오9 – 학생 I가 문자열과 반복문 등 여러 개념을 동시에 적용해야하는 문제가 어려운 경우**

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

학생 I는 문자열 중 단어의 개수를 세는 문제를 풀기로 하였습니다.

텍스트, 스크린샷이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.문자가 연속해서 나타나지 않는지 검사하는 코드를 아래와 같이 작성하였습니다.

하지만 해당 코드는 Runtime Error가 계속 발생하였고, 어디가 문제인지 파악하지 못한 학생 I는 피드백을 요청했습니다.

‘’’

***Runtime Error***

***입력 파트 (O)***

*문제없습니다!*

***문제풀이부분1 (X)***

*현재 abc[26] 배열을 for문을 이용해 길이(len)만큼만 초기화하고 있습니다. 그러나 abc 배열은 알파벳 26개를 모두 나타내는 배열이므로, 항상 26개 전체를 0으로 초기화해야합니다.*

***문제풀이부분2 (X)***

*또한, 현재 코드에서는 같은 문자가 다시 나타나면 word=0;을 설정하여 그룹 단어가 아니라고 판단하는데, 이를 체크하는 순서가 잘못되었습니다.*

*불필요하게 abc 값을 변경할 가능성이 있기 때문에, 이미 등장한 문자인데 연속되지 않았다는 사실이 확실할 경우 바로 word 변수값을 설정해야합니다.*

*해당 부분을 고려하여 다시 작성해보세요!*

***출력 파트 (O)***

*문제없습니다!*

‘’’

위와 같은 피드백을 받은 학생 I는 피드백에 따라 두 가지 잘못된 점을 수정하여 아래와 같은 코드를 작성하였습니다.

텍스트, 스크린샷이(가) 표시된 사진

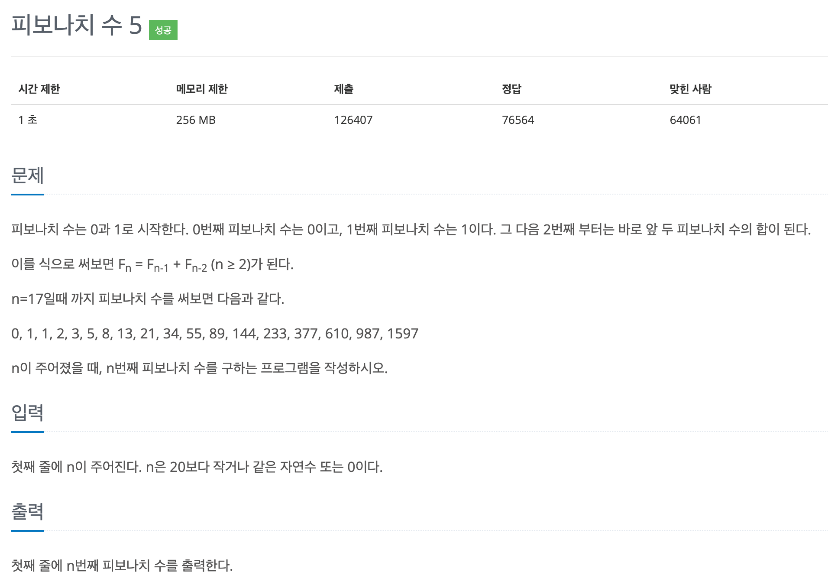
AI가 생성한 콘텐츠는 부정확할 수 있습니다.

abc의 배열 변수 선언 부분과 연속인지 아닌지를 판단하는 조건문을 다시 작성하여 제출한 결과, OJ에서 Accepted를 받게 되었습니다.



**3-10. 시나리오10 – 논리는 알지만 구현을 실수한 경우**

학생 J가 숙제로 다음문제를 풀기로 하였습니다.

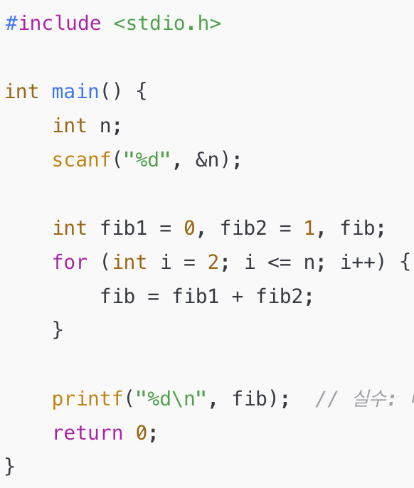


학생은 고등학교때 피보나치 수열을 공부한적이 있어 논리를 알고있습니다.

f(n)=f(n-1)+f(n-2)

f(1)=1, f(2)=1.

하지만 구현의 어려움이 있어 다음과 같은 코드를 제출하였습니다.



학생은 “fib = fib1 + fib2;로 계산했는데 왜 틀린 거지?”라고 생각을 하였습니다. 구현을 할 수 없던 학생은 피드백 버튼을 눌렀습니다.

“””

***Wrong Answer***

**입력 파트 (O)**

입력받는 부분에서 문제는 없습니다. 사용자에게 값을 정상적으로 입력받고 있습니다

**문제풀이 부분 (X)**

현재 fib1과 fib2 값을 이용하여 피보나치 수를 계산하고 있지만, 반복문을 도는 동안 fib1과 fib2의 값을 갱신하지 않고 있습니다.  
 이로 인해 올바른 피보나치 수열이 생성되지 않고, 마지막 계산된 fib 값도 원하는 결과와 다를 수 있습니다.  
 이전 값을 기억하면서 갱신하는 방법에 대해 더 알고 싶다면, 반복문을 사용한 피보나치 수열 구현 방법을 확인해보세요!

**출력 파트 (X)**

마지막으로 출력하는 fib 값이 올바른 피보나치 수열의 n번째 값이 아닐 수 있습니다.  
이유는 fib1과 fib2를 갱신하지 않아서 fib 변수에 기대했던 값이 저장되지 않기 때문입니다.  
이전 두 값을 계속 업데이트하며 fib를 계산하고, 최종적으로 fib2를 출력하도록 수정하면 올바른 결과를 얻을 수 있습니다.

“””

학생J는 이전값들을 갱신해야한다는 것을 깨닫고 다시 코드를 작성하여 어떻게 구현해야하는지을이해 하였습니다.

**3-11. 시나리오11 – 학생 K가 예외 조건을 설정하지 못한 경우**

텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.학생 K가 아래와 같은 알고리즘 문제를 풀기 위해 코드를 작성하였다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.학생 K는 문제에서 제시된 것처럼 가장 큰 자연수를 찾기 위해 while문을 이용해서 코드를 작성하였다.

그러나 OJ사이트에서 Wrong Answer 결과가 나왔고, break 조건을 어떻게 세워야할지 몰라서 피드백 요청 버튼을 클릭하였다.

‘’’

***Wrong Answer***

***입력 파트 (O)***

*문제없습니다!*

***문제풀이부분1 (X)***

*현재 조건이 b가 a보다 크거나 같음(>=)으로 설정되어있는데, 자연수의 합이 S보다 넘어가는 경우까지 확인하게 됩니다. 따라서 a와 b가 같은 경우에 대한 예외 처리가 필요합니다!*

*해당 부분을 다시 고려하여 작성해보세요!*

***출력 파트 (O)***

*문제없습니다!*

‘’’

학생 K는 위의 피드백을 참고하여 a와 b가 같은 경우에 대한 예외를 처리하는 코드를 추가로 작성하였습니다.

아래와 같이 수정된 코드를 작성한 뒤, 제출하여 Accepted를 받았습니다!

텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.



**3-12. 시나리오12 – 학생 O가 문제를 풀었으나, 스스로 더 점검하고 싶어하는 경우**

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

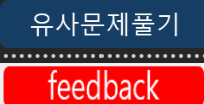
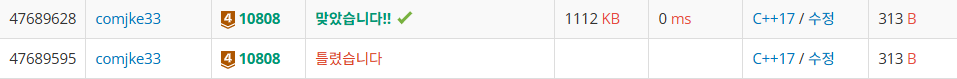
AI가 생성한 콘텐츠는 부정확할 수 있습니다.

학생 O는 이번주 이론 시간에 문자열을 배우면서 OJ 시스템에 올라온 문제를 풀었다.

위와 같은 문제를 풀면서 문자열을 반복문으로 접근하여 처리하는 방식에 대해서 이해했다. 여러 번의 수정 끝에 코드를 작성하고 힘들게 Accepted를 받은 학생 O는 자신감이 붙어서 문자열을 다루는 추가적인 문제를 풀고 싶어졌다.

텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.따라서 Accepted 옆의 유사 문제 풀기 버튼을 클릭하였더니, 아래와 같은 사이트로 연결되었다.

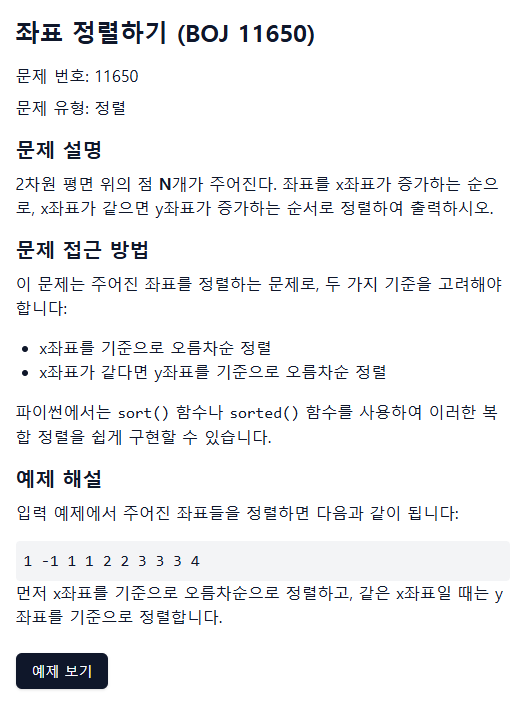


해당 페이지는 백준의 ‘알파벳 개수’라는 문제 페이지였고, 이 문제 또한 문자열을 처리하는 문제라서 비교적 친숙했다. 실제로 해당 문제를 풀면서 문자열을 반복문으로 접근하는 방식에 대해서 잘 숙지하게 되었고, 여러 번의 시도 끝에 백준 사이트에서도 통과를 받았다.

**3-13. 시나리오13 – 학생 P가 OJ기능을 이용해 문제를 풀었으나 이용하지 않고 문제를 풀 수 있는지 확인하고 싶은 경우**



학생 P는 이 문제를 풀고 싶었으나 문제에서 무엇을 요구하는지 이해하기 힘들어 마크업 기능을 사용했다.

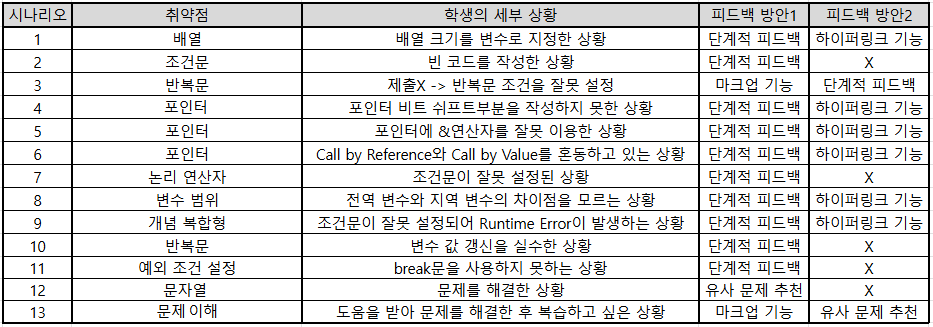


마크업 기능을 사용하여 문제를 풀 수 있었던 P는 이와 비슷한 문제를 마크업 기능을 사용하지 않고 풀어보고 싶었다.



P는 유사문제풀기 버튼을 눌러 유사한 문제가 있는 사이트로 이동하였고, 이번에는 마크업 기능을 사용하지 않고 풀어보며 복습하게 되었다.

**4. 요구사항**



* 내가 제출한 코드에 대해 피드백을 받고 싶다
  + **학생이 제출한 코드에 대해 단계적 피드백 기능 제공**
* 포인터와 같은 어려운 내용은 탄탄한 기초 개념습득이 필요하다
  + **기초 개념 학습을 위한 하이퍼링크 기능 제공**
* 문자열 처리 등 다양한 문제에 대해 유사한 문제를 풀어볼 수 있으면 좋겠다
  + **백준 OJ 기반 유사 문제 추천 기능 제공**
* 문제를 이해하지 못하는 경우 나의 코딩 실력과 상관없이 문제를 풀 수 없다
  + **마크업 기능 제공**

**5. 기능 정의**

**[1] 기본개념 하이퍼링크 기능**

* **목표**: 학생들이 자주 발생하는 문법 실수를 점검하고, 스스로 학습할 수 있도록 돕습니다.
* **기능**: 학생들이 필요한 문법을 신속하게 학습할 수 있도록 관련 개념을 제공하는 웹사이트로 하이퍼링크를 제공합니다. 이를 통해 학생들이 반복적으로 문법을 복습하고, 자주 발생하는 실수를 예방할 수 있게 지원합니다.

**[2] 문제 이해를 위한 마크업 기능**

* **목표**: 학생들이 문제를 정확히 이해하고, 문제 해결에 필요한 정보를 파악할 수 있도록 돕습니다.
* **기능**: 문제 설명을 이해하기 어려운 학생들을 위해, 요구사항을 더 명확하고 자세히 전달할 수 있도록 마크업 기능을 활용하여 문제를 쉽게 이해하고 해결할 수 있도록 지원합니다

**[3] 단계별 피드백 기능**

* **목표**: 학생들이 문제를 이해해도 구현에 어려움을 겪는 문제를 해결할 수 있도록 돕습니다. 학생들이 문제를 단계적으로 이해하고, 알고리즘을 체계적으로 접근하여 문제를 해결할 수 있도록 유도합니다.
* **기능**: 학생들이 제출한 코드에 대한 피드백을 받아볼 수 있습니다. 이를 통해 학생들이 자신의 코드에서 무엇이 문제였는지 스스로 생각해볼 수 있습니다.

**[4] 유사 문제 추천 기능**

* **목표:** 학생들이 문제의 정답을 맞힌 후에도 유사한 문제를 풀어보며 복습의 기회를 제공합니다.
* **기능:** 학생들이 문제를 복습할 수 있도록 그와 유사한 문제를 백준 OJ에서 찾아 줍니다. 이를 통해 학생들은 자신의 실력을 다시 점검해볼 수 있으며, 자연스럽게 복습할 수 있습니다.

**6. 기능 세부 정의**

**[1] 기본개념 하이퍼링크 기능**

* 작성된 피드백에서 개념, 문법적인 오류에 대한 피드백이 있다면 해당 내용과 가장 유사한 **개념서 사이트**로 이동할 수 있는 기능을 제공합니다.
  + C 언어 개념서 링크 제공
    - C 언어의 주요 개념(변수, 포인터, 구조체, 배열, 문자열)을 학습할 수 있는 개념서 링크를 제공합니다.
* 평가 방법
  + 문법 오류를 포함한 샘플 코드 20개를 제작하고 라벨링
  + 라벨링된 개념과 해당 기능을 사용하여 연결된 개념이 일치하는 성공률이 80% 이상 성공

**[2] 마크업 기능**

* 문제 속 개념을 토글 형태로 설명하고 중요한 키워드에 강조 표시
  + 문제 풀이 중 관련 개념을 확인할 수 있도록 키워드를 찾아 강조합니다.
  + 문제 설명 내에 핵심 키워드에 밑줄 표시 및 볼드체 처리를 해줍니다.
* 평가방법
  + 핵심 개념을 라벨링한 알고리즘성 문제 20개 수집하여 데이터셋 구축 후, 마크업 기능이 각 문제 속 핵심 개념을 알맞게 골라 힌트를 제공한 성공률이 80% 이상

**[3] 단계별 피드백 기능**

* 제출된 코드의 구현율을 기반으로 다른 정도의 피드백을 제공합니다.
  + 30% 미만 구현율의 [1단계]
    - 핵심 개념을 전체적으로 제공하면서, 기초적인 문법에 대한 설명도 제공합니다.
* 31%~80% 구현율의 [2단계]
* 부족한 개념을 일부분 제공하면서, 스스로 해결을 위해 질문의 형태로 피드백을 제공합니다.
* 81% 이상 구현율의 [3단계]
* 코드 오류의 원인을 스스로 해결할 수 있도록 질문 형태로 피드백을 제공합니다.
* 입력파트, 문제풀이파트(1~N), 출력파트로 단계를 나누어 피드백을 제공합니다.
  + 입력파트
    - 주로 초심자 학생들을 위한 파트로, 변수명부터 입력받는 방법등에 대해 자세한 피드백을 제공합니다.
* 문제풀이파트(1~N)
* 문제풀이를 단계적으로 나누어 학생이 코드의 흐름을 명확히 파악할 수 있게 피드백을 제공합니다.
* 출력파트
  + - * 문제에서 요구하는 공백 처리, 줄바꿈에 대한 피드백을 제공합니다.
  + 평가방법
    - 오류 원인이 라벨링되어있는 샘플 코드 데이터셋 20개를 제작하여 단계적 피드백 기능 사용 시, 라벨링된 오류에 피드백이 주어지는지 확인.
      * Macro-average Recall > 80%

**[4] 유사 문제 추천 기능**

* 학생이 제출해서 푼 문제에 대해 유사문제를 제공합니다.
* 학생은 문제를 풀고 정답을 확인합니다.
* 학생이 제출한 코드가 정답일 경우에만 피드백 버튼 대신 유사문제 추천버튼을 눌러 새로운 문제로 이동할 수 있습니다.
* 방금 해결한 문제와 구조가 유사한 문제로 이동합니다.
* 평가방법
  + 자체 제작된 C언어 기초 문제 데이터셋 50개를 대상으로 선정된 유사문제가, 기존 OJ 문제와의 카테고리가 일치하는지 여부를 판단함. 성공률 90% 이상

**7. 세부 기능별 개발 및 구현 계획**

**[1] 기본 개념 하이퍼링크 기능**

* **기본 개념 사이트 링크 DB 구축**
  + 사전에 정의된 키워드 5개(변수, 조건문, 반복문, 배열, 포인터)를 기반으로 개념 설명 URL과 매핑하여 데이터베이스를 구축
  + MySQL에 [키워드], [URL] 형식으로 데이터를 저장
* **링크 안내 텍스트 생성**
  + LLM이 생성한 피드백을 파싱하여 문법 키워드를 추출
  + 중복 호출된 링크가 있다면 1번만 표시

**[2] 마크업 기능**

* **마크업 페이지 생성**
  + 관리자가 문제를 등록하면, LLM을 통해 문제 설명을 분석 후, 자동으로 마크다운 형식의 페이지를 생성
* **마크업 페이지 저장 및 활용**
  + 생성된 마크업 HTML 파일은 로컬에 저장되어, 마크업 버튼의 값에 따라 자동으로 변환

**[3] 단계별 피드백 기능**

* **구현율 계산 기준**
  + 각 코드의 구현 여부는 아래 5가지의 평가 기준에 따라서 구현율을 계산
    - 변수 선언과 초기화를 올바르게 했는가? (20%)
    - 조건문과 반복문을 적절하게 사용했는가? (20%)
    - 문제에서 요구한 기능을 정확하게 구현했는가? (20%)
    - 입출력 결과가 문제 예제와 일치하는가? (20%)
    - 문제 해결을 위한 알고리즘 설계가 논리적인가? (20%)
* **구현율 단계 정의**
  + 구현율 0% ~ 30% (1단계): 학생이 코드의 대부분을 작성하지 않은 경우.
    - 피드백 내용
    - 문제를 해결하는 데 필요한 기본 개념을 제공.
      * 기초적인 문법에 대한 설명 및 예시 제공.
  + 구현율 31% ~ 80% (2단계): 학생이 문제 해결의 일부를 시도한 경우.
    - 피드백 내용
      * 부족한 부분에 대해 질문 형태로 피드백 제공.
      * 부분적으로 해결한 문제에 대한 개선 방향 제공.
  + 구현율 81% 이상 (3단계): 코드의 주요 부분이 완성되었으나, 오류가 있을 때.
    - 피드백 내용
      * 코드 오류를 스스로 해결할 수 있도록 질문 형식으로 제공.
      * 각 파트별로 입력 파트, 문제 풀이 파트, 출력 파트에 대해 단계별로 피드백을 구분하여 제공.
* **피드백 생성**
* 구현율 단계에 따라 다른 프롬프트를 적용하여 피드백 생성

**[4] 유사문제 추천 기능**

* **문제 데이터 수집**
  + 문제 태그 데이터 설계
    - 문제 데이터베이스(Problem Table)에 tags 필드를 추가하여 여러 개의 태그를 저장할 수 있도록 함.
  + 관리자 페이지에서 문제 태깅 기능 추가
    - 문제 등록 시, 관리자가 관련 태그를 선택하여 저장할 수 있도록 하는 기능 제공.
* **문제 추천 로직 구현**
  + 사용자가 “Accepted”받은 문제의 태그를 확인.
  + 백준 OJ에서 동일한 태그를 가진 문제를 검색하여 해당 문제의 링크 생성.
* **프론트엔드와 연동**

**결론**

이에 따라, 저희는 학생들의 코드 제출 내역을 기반으로 **개인별 맞춤 피드백을 제공하는 LLM 기반 OJ 시스템**을 개발하고자 합니다. 단순히 정답 여부를 판단하는 것을 넘어, 학생 개개인의 코드를 분석하고, 취약한 개념을 보완할 수 있도록 돕는 피드백 시스템을 구축하는 것이 목표입니다.

이 시스템은 학생들의 코드에서 논리적 오류를 분석하고, 해결 방법을 제안하는 기능을 포함하며, 특정 개념을 놓쳤을 경우 이를 보완할 수 있도록 관련 개념 학습 자료를 연결하는 기능을 제공합니다. 궁극적으로, 학생들이 C언어의 개념을 깊이 이해하고, 문제 해결 능력을 체계적으로 향상시킬 수 있도록 지원하는 것이 본 프로젝트의 핵심입니다.