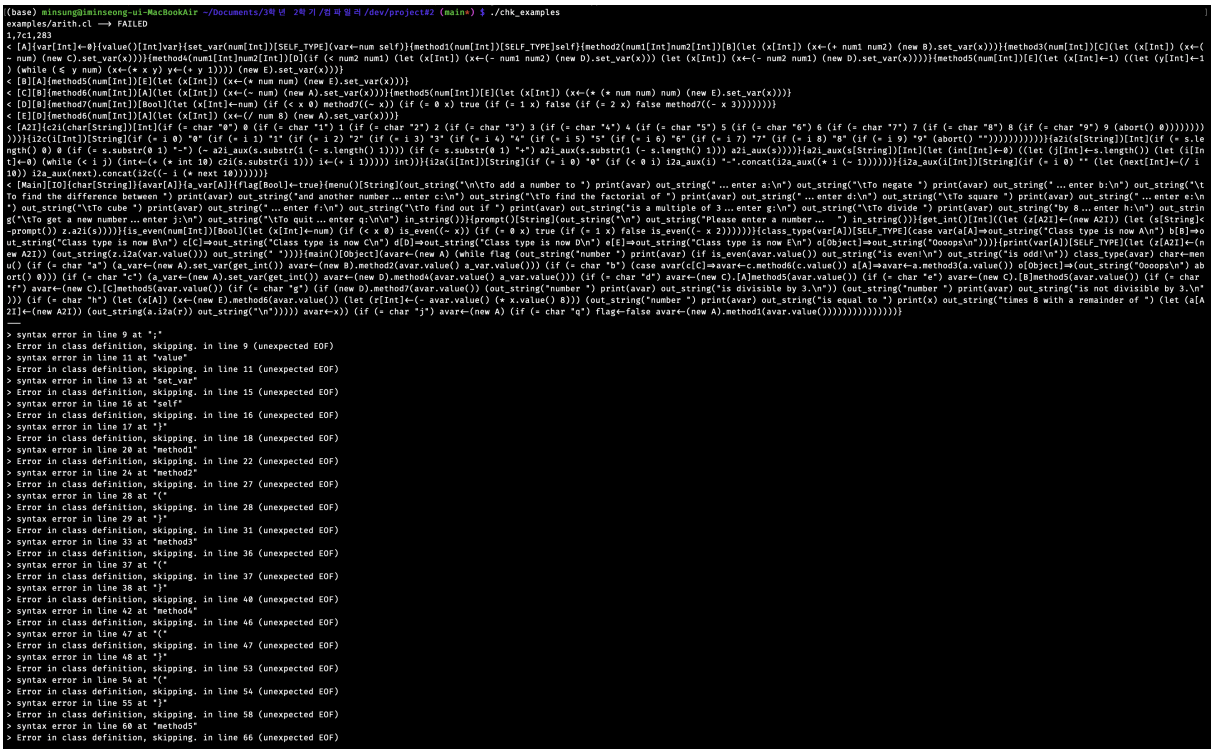
컴파일러 Project#2

컴파일(make)과 실행(chk_examples) 과정을 보여주는 화면 캡처

```
(base) minsung@iminseong-ui-MacBookAir ~/Documents/3학 년  2학 기 /컴 파 일 러 /dev/project#2 (main*) $ make
bison -d cool.y
flex cool.l
gcc -Wall -O3 -c lex.yy.c
gcc -Wall -O3 -c cool.tab.c
gcc -Wall -O3 -c node.c
gcc -o cool_parser lex.yy.o cool.tab.o node.o -ll -mmacosx-version-min=13.3
```



실행 결과에 대한 설명, 소감, 문제점

- 실행 결과 설명
  1. **bad.cl의 결과**
     - 구문 오류가 다수 발생하였으며, 대부분 `unexpected EOF` 또는 특정 토큰(`;`, `test1`, `C` 등)에서 발생한 오류입니다.
     - 오류 메시지는 `syntax error in line X at <토큰>` 형식으로 표시되며, 이어지는 메시지로 "Error in class definition, skipping."이 출력됩니다.
     - 최종적으로 **36개의 오류**가 발견되었다고 출력되었습니다.
  2. **good.cl의 결과**
     - 오류 메시지 패턴은 `bad.cl`과 유사하며, 특정 연산자(`~`, `+`, `@`) 또는 변수 이름(`y`, `z`, `foo`)에서 구문 오류가 발생하였습니다.
     - 결과적으로 추상구문트리(AST)를 출력하지 못했으며, **54개의 오류**가 발견되었다고 출력되었습니다.

- 소감
  - 이번 과제는 COOL 언어의 문법을 BNF로 정의하고 구문 검사 및 AST 생성을 구현하는 과제였습니다.
  - **엄청난 노력과 시간을 쏟아부었음에도 실행 결과가 기대에 미치지 못하여 아쉬움이 큽니다.**
  - Bison과 Flex를 사용한 파서 작성은 처음이었고, 이를 통해 구문 분석기의 내부 작동 방식을 배우는 좋은 기회였습니다.
  - 하지만 많은 오류가 발생하면서 구현의 복잡함과 디버깅의 중요성을 다시 한번 느꼈습니다.
- 문제점 분석
  1. **BNF 정의의 불완전성**
     - 구문 오류가 지나치게 많이 발생한 이유는 BNF 정의가 COOL 언어의 문법을 정확히 반영하지 못했기 때문일 가능성이 큽니다.
     - 예를 들어, 클래스 정의, 조건문, 연산자 우선순위와 같은 문법 규칙이 누락되었거나 잘못 작성되었을 수 있습니다.
  2. **에러 핸들링 부족**
     - 오류가 발생했을 때 오류 토큰만 출력하고 이후 처리가 제대로 이루어지지 않는 문제가 있습니다.
     - `unexpected EOF`와 같은 메시지가 반복적으로 출력된 이유는 구문 오류 후 복구 과정이 미흡했기 때문입니다.
  3. **추상구문트리(AST) 생성 실패**
     - `good.cl` 파일의 AST가 출력되지 않은 이유는 의미 행위(semantic actions)가 제대로 정의되지 않았기 때문일 수 있습니다.
     - `node.h`와 `node.c`에서 정의된 노드 생성 관련 코드에 문제가 있거나, Bison의 의미 행위 연동이 실패했을 가능성이 있습니다.
  4. **디버깅 도구 부족**
     - Bison에서 제공하는 디버깅 옵션(`-debug` 등)을 활용하지 않아 구문 분석 과정을 시각적으로 확인하지 못했을 수 있습니다.
- 개선 방안
  1. **문법 정의 재검토**
     - COOL 언어의 문법 정의를 다시 확인하고, BNF 정의가 정확히 반영되었는지 검토해야 합니다.
     - 공식 COOL 문법 또는 수업 자료와 비교하여 누락된 규칙이나 잘못된 정의를 수정해야 합니다.
  2. **디버깅 도구 활용**
     - Bison의 디버깅 옵션을 활성화하여 토큰 처리 및 구문 분석 과정에서 문제가 발생하는 부분을 정확히 파악해야 합니다.
     - `yydebug`를 사용하면 상태 전환 및 처리 과정을 확인할 수 있습니다.
  3. **의미 행위(semantic actions) 개선**
     - 의미 행위 정의가 구문 분석과 올바르게 연동되었는지 확인하고, `node.h` 및 `node.c`에서 AST 노드 생성을 테스트해야 합니다.
  4. **에러 복구 로직 추가**
     - 구문 오류 이후 복구 로직을 추가하여 오류를 무시하고 다음 구문을 처리하도록 개선해야 합니다.

- 이를 통해 `unexpected EOF` 오류를 줄이고 보다 정확한 오류 메시지를 출력할 수 있습니다.

5. **테스트 케이스 확장**
    - 다양한 COOL 프로그램을 테스트하여 파서가 예상대로 동작하는지 확인하고, 발견된 문제를 하나씩 해결해야 합니다.

- 최종 결론

  이번 과제는 파서 작성의 복잡성을 실감하는 동시에, 구문 분석기와 AST 생성에 대한 깊은 이해를 요구하는 도전적인 과제였습니다.

  결과적으로 **목표를 달성하지 못했지만**, 이를 통해 파서 작성의 기본 개념을 학습할 수 있었으며, 부족한 부분을 보완하여 더 나은 결과를 기대할 수 있을 것입니다. **향후 개선 방향을 기반으로 지속적으로 디버깅하고 수정해 나가겠습니다.**