# 프로젝트 **#2**
# **Parser** 구현

# 컴파일러 과제 **#2 - Parser**

- C- Parser 작성 – recursive descent parser
  - appendix A.2의 C- grammar 참조. (첨부 grammar.pdf참조)
  - 구현환경: scanner와 동일
  - 입력
    - sample 프로그램 2개(scanner때 사용했던 프로그램 2개)
    - 그 외 에러를 포함한 임의의 C- program으로 테스트함.
  - 출력
    - 교재182페이지 TINY fig. 4.10(다음 페이지 참조) 참조하여 본인이 재구성할 것
  - 오류처리: TINY 언어의 오류처리 참고

# Fig.4.10 Display of a TINY Syntax tree by the printTree procedure

```
Read: x
If
  Op: <
    const: 0
    Id: x
  Assign to: fact
    const: 1
  Repeat
    Assign to: fact
      Op: *
        Id: fact
        Id: x
    Assign to: x
      Op: -
        Id: x
        const: 1
    Op: =
      Id: x
      const: 0
  Write
    Id: fact
```

- 기한
  - 프로그램: 5월 26일 자정까지 LMS과제게시판에 업로드
  - 보고서:
    - 제출장소: 공대9호관 417-2호 앞 상자
    - 제출일: 5월 26일, 27일(9시~19시), 5월 28일(9시~22시)

# 보고서 양식

- 1. 문제에 대한 설명:
  - Grammar를 recursive descent parser로 구현하기 위해 처리한 사전작업(EBNF 변경) 설명: 각 rule에 대해 변경했을 경우, 변경 전 및 변경 후를 명시함. 변경되지 않았으면, 변경 전 변경 후 동일하다고 명시
- 2. syntax tree structure for C-
  - 본인이 적용한 AST에 대해 설명(교재 136-137 혹은 3장 강의노트 73~76페이지 참조)
- 3. 프로그램에 대한 설명
  - function 설명 (입력, 출력, 목적)
  - 예제 수행
    - 두 샘플프로그램(1.c, 2.c)의 실행결과 첨부 (사이즈 작아도 관계없음. 확인가능하기만 하면 됨)
  - 1.c의 AST(tree형태 그림)

- 기타사항
  - 구현언어 및 환경(scanner와 동일)
  - 테스트
    - 스캐너와 동일:
      - c:>filename inputFile outputFile: input 파일에 적용한 결과가 output 파일로 나오도록
      - (예) parse a.c a.txt
  - 토큰입력을 받으려면 이전 과제(scanner)를 이용해야 할 것임

# 채점기준

- 보고서
  - 빠뜨린 항목이 있으면 감점
  - 내용 충실도에 따라 가,감점
- 프로그램
  - 컴파일이 안되면 0점(보고서 점수도 없음),
  - 두 샘플 프로그램(1.c, 2.c)에 대해 실행한 결과가 틀리면 감점 및 틀린 정도에 따라 보고서 점수도 감점
    - 두 샘플 프로그램에 대해 정상적으로 결과가 나와야 파서 구현이 제대로 되었다고 판단.
  - 에러처리가 제대로 되지 않으면 감점
    - 에러 프로그램은 임의로 구성함.