Project 2

Due: 10/14/2024 10:30 AM

서론

(1) 각 프로젝트의 파일은 Project 2 -> Problem <번호> 폴더 구조로 구성하고, 각 Problem 에서 요구하는 모든 파일을 저장하세요. Project 2 폴더를 압축(zip)하고 압축 파일 이름은 학번으로 하여 제출하세요.
(2) 보고서가 필요한 경우는 pdf 형식으로 제출하며, 언어는 영어/한글 모두 무방합니다.
(3) 프로젝트의 deadine 까지 blackboard 의 <과제 및 시험>란에 제출하세요.
(4) 프로젝트의 마감기한을 지키지 못한 경우, 감점 없이 0 점 처리할 것입니다.

■ 모든 프로젝트는 개인별 프로젝트입니다.

문제 1(10 점/150 점): Introduction to Bison and Parser

제시된 프로그램을 설치하고, 지시문에 따라 필요한 파일을 제출하세요.

- Bison(Yacc)은 C code 를 parsing 하기 위해 사용하는 tool 입니다. 문제 1 은 Bison 을 활용한 parsing 이 어떻게 이루어지는지 살펴보고 학습할 수 있도록 준비하였습니다. 그리고, 제공한 코드를 모두 잘 살펴보고 전체적인 흐름과 논리를 이해해보기 바랍니다.

① Linux 터미널에서 "sudo apt-get install bison" command 를 입력하여 bison 도구를 설치한 뒤, 제공된 "bison.sh" script 를 실행하여 생성된 file 을 캡처한 이미지 (problem1_1.jpg)(5 점)

```
compiler@ubuntu:~/work/Project_2$ ll
total 144
drwxrwxr-x 2 compiler compiler  4096 Sep  6 05:47 ./
drwxr-xr-x 6 compiler compiler  4096 Sep  6 05:38 ../
-rw-rw-r-- 1 compiler compiler    76 Sep  6 05:42 bison.sh
-rw-rw-r-- 1 compiler compiler   276 Sep  6 05:42 example1.l
-rw-rw-r-- 1 compiler compiler   543 Sep  6 05:47 example1.y
-rw-rw-r-- 1 compiler compiler 45744 Sep  6 05:47 lex.yy.c
-rwxrwxr-x 1 compiler compiler 28336 Sep  6 05:47 project2_example1*
-rw-rw-r-- 1 compiler compiler 43512 Sep  6 05:47 y.tab.c
-rw-rw-r-- 1 compiler compiler  2245 Sep  6 05:47 y.tab.h
```

② "project2_example1" executable file 을 통해 세 입력(heat on, target temperature 40, target temp 33)을 parsing 한 결과를 캡처한 이미지 (problem1_2.jpg)(5 점)

```
compiler@ubuntu:~/work/Project_2$ ./project2_example1
heat on
        Heat turned on or off
target temperature 40
        Temperature set
target temp 33
error: syntax error
```

문제 2(50점/150점): Basic Data Structure for Parse Tree

구현하고자 하는 parse tree는 그림 1과 같이 doubly linked list 구조를 갖고, tree의 각 node는 그림 2와 같이 정의합니다. Parent는 단 하나의 child node를 가질 수 있으며, 같은 tree depth 에 있는 sibling node들은 prev와 next pointer를 이용하여 연결됩니다. 제시한 NODE 구조에서 name은 node의 이름을 정의하는 char *입니다.

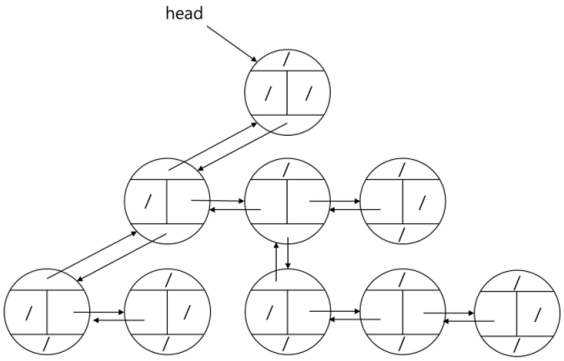


그림 1. Parse tree의 구조 예시.

Struct NODE

name	parent	child	prev	next

그림 2. Parse tree의 node의 데이터 구조.

아래의 제시문을 읽고, 조건을 만족하는 코드를 node.c에 작성하여 제출하고, 구현을 검증한 내용 또한 제출하기 바랍니다. C standard library를 제외한 library는 사용할 수 없습니다.

① node의 구조체 NODE를 선언(5점)

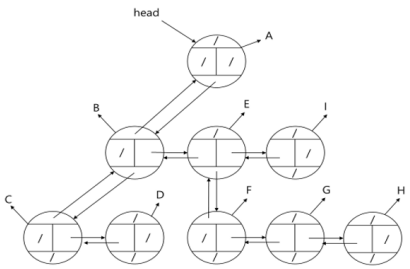
② char* name을 이름으로 하는 NODE를 생성하는 함수 MakeNode(char* name)을 작성(5점)

③ Parse tree에서 parent node에 child node(this node)를 insert 하는 함수 InsertChild (NODE* parent_node, NODE* this_node)를 작성(10점)

④ Parse tree에서 같은 depth에 있는 sibling node(prev_node)에 새로운 node(this node)를 insert 하는 함수 InsertSibling(NODE* prev_node, NODE* this_node)를 작성(10점)

⑤ Parse tree의 특정 node를 root로 하는 subtree를 DFS 순서로 char* name을 출력하는 함수 WalkTree(NODE *node)를 작성(10점)

⑥ node.c의 구현 검증: 그림 3과 같은 parse tree를 생성하고 출력하기(problem2.jpg)(10점)




```
compiler@ubuntu:~/work/Project_2/problem2$ ./problem2
(A
(B
(C)
(D))
(E
(F)
(G)
(H))
(I))compiler@ubuntu:~/work/Project_2/problem2$
```

그림 3. 예제 parse tree와 이를 DFS 순서로 출력한 모습.

문제 3(90점/150점): Building Parse Tree

제공된 context-free-grammar(project3.y)와 tokenizer(project3.l), 문제 2에서 작성한 node.c 를 이용해 문제에서 제시한 mat_mul.c를 parsing하고, parse tree를 build하기 바랍니다. 그림 4는 mat_mul.c 를 parsing 하여 얻은 parse tree의 일부를 표현한 것입니다.

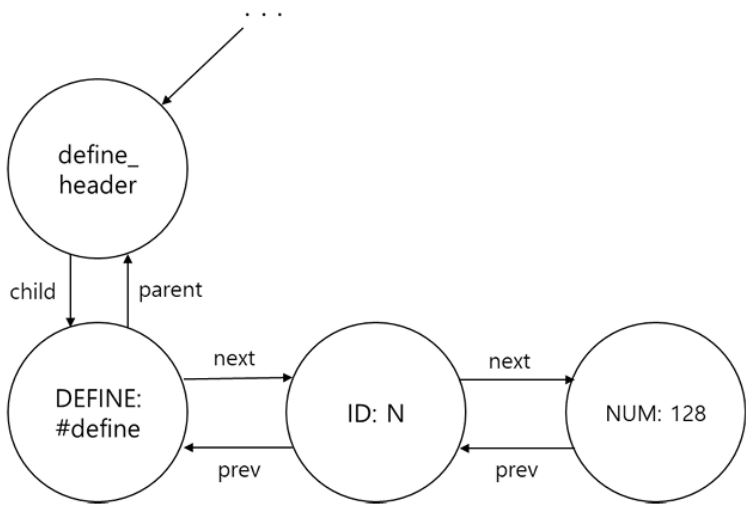


그림 4. mat_mul.c에 대한 parse tree의 일부분(define_header).

Context-free-grammar 상의 non-terminal과 terminal은 parse tree 의 node에 해당합니다. Grammar rule에 따라 non-terminal 을 derivation 할 때, action을 통해 node를 생성하고 parse tree를 생성하게 됩니다. 이때, grammar의 sentential form에서 첫번째 non-terminal /terminal이 child node가 되고, 이외의 nonterminal/terminal들은 sibling node가 됩니다.

Bison은 bottom-up parsing 을 사용하므로, 그림 4의 경우에는 'DEFINE', 'ID', 그리고 'NUM' 의 node 를 생성하여 sibling으로 연결한 후 'define_header' node를 생성하여 parent-child 관계로 연결하는 방법으로 parse tree를 생성하였습니다. 이를 참고하여 mat_mul.c에 대한 parse tree를 생성하면 되겠습니다.

생성한 parse tree는 문제 2에서 작성한 WalkTree(NODE *node) 함수를 통해서 DFS 순서로 출력해야 합니다. 이때, 가독성을 위해 다음 조건을 만족시키기 바랍니다:

- Non-terminal/terminal에 대한 derivation이 시작되면 '('를 열기
- 해당 non-terminal/terminal에 대한 derivation이 끝나면 ')'를 닫기

아래 제시문을 읽고, 이에 따라 제출물을 준비하기 바랍니다.

① Parse tree를 생성하고 출력하는 코드가 포함된 project2.y와 project2.l, 그리고 이를 컴파일하기 위한 shell script(project2.sh)(40점)

- project2.l은 별도 수정 및 보완 없이 그대로 사용하여도 됩니다. project2.y에는

parsing에 필요한 grammar rule을 작성해두었습니다. 이를 참고하여 필요한 부분을 마저 작성하면 됩니다.

② project2.y의 구현에 대한 설명이 포함된 보고서(project2.pdf)(30점)

③ Parse tree를 출력하여 output.txt와 동일한 출력이 나온 사진(problem3.jpg)(20점)