Project 3

Due: 11/1/2023 10:30 AM

(1) 각 프로젝트의 파일은 Project <번호> -> Problem <번호> 폴더 구조로 구성한다. 예를 들어, Project 3->Problem 1 의 폴더에 Problem 1 에서 요구하는 모든 파일을 저장한다. 모든 파일은 Project 3 하에서 압축(zip)하고 **압축 파일 이름은 학번**으로 한다.

(2) 보고서가 필요한 경우는 pdf 형식으로 제출한다.

(3) 프로젝트 제출일의 수업시작(오전 10 시 30 분) 전까지 blackboard 를 통하여 제출한다.

■ **모든 프로젝트는 개인별 프로젝트입니다.**

문제 1(60 점): Building basics for parse trees

구현하고자 하는 parse tree 는 그림 1 과 같이 doubly linked list 구조를 갖고, tree 의 각 node 는 그림 2 로 정의한다. Parent 는 단 하나의 child node 를 가질 수 있으며, 같은 tree depth 에 있는 sibling node 들은 prev 와 next pointer 를 이용하여 연결한다. 제시한 NODE 구조에서 name 은 node 의 이름을 정의하는 char *이다.

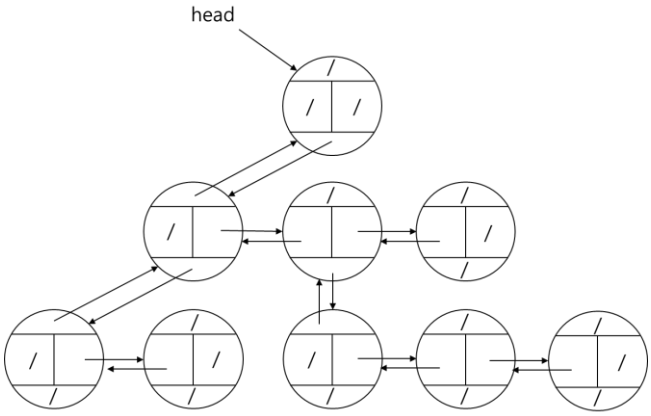


그림 1. Parse tree 의 구조

Struct NODE

name	parent	child	prev	next

그림 2. Parse tree node 의 구조

이를 바탕으로 다음의 코드를 node.c 에 작성하여 제출한다. C standard library 를 제외한 library 는 사용할 수 없다.

① (5 점) Node 의 구조체 NODE 를 선언하시오.

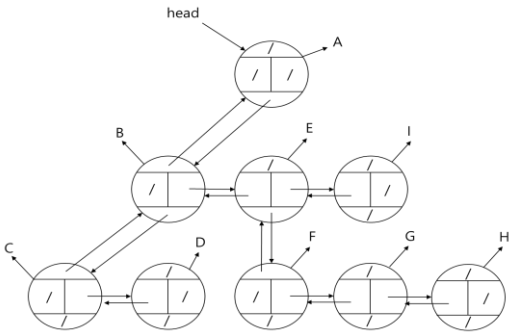
② (10 점) char* name 을 받아 NODE 를 생성하는 함수 MakeNode(char *name)를 작성하시오.

③ (10 점) Tree 에서 parent node 에 child node(this node)를 insert 하는 함수 void InsertChild(NODE* parent_node, NODE* this_node)를 작성하시오.

④ (10 점) Tree 에서 같은 depth 에 있는 sibling node(prev_node)에 새로운 node(this node)를 insert 하는 함수 void InsertSibling(NODE* prev_node, NODE* this_node)를 작성하시오.

⑤ (10 점) Tree 의 어떤 node 를 root 로 하는 subtree 를 DFS 순서로 char* name 을 출력하는 void WalkTree(NODE *node)의 함수를 작성하시오.

⑥ (15 점) 아래와 같은 tree 를 위의 함수들을 이용하여 구현하고 출력하시오.




```
compiler@ubuntu:~/work/Project_3$ ./problem1

(A
(B
(C)
(D))
(E
(F)
(G)
(H))
(I))compiler@ubuntu:~/work/Project_3$
```

그림 3. 예제 parse tree 와 출력 example.png

문제 2(150 점): Building Parse Tree

제공된 context-free-grammar(project3.y)와 tokenizer(project3.l), 문제 1 에서 작성한 node.c 를 이용해 문제에서 제시한 mat_mul.c 를 parsing 하고, parse tree 를 build 한다. 그림 4 는 mat_mul.c 를 parsing 하여 얻은 parse tree 의 일부를 표현한 것이다.

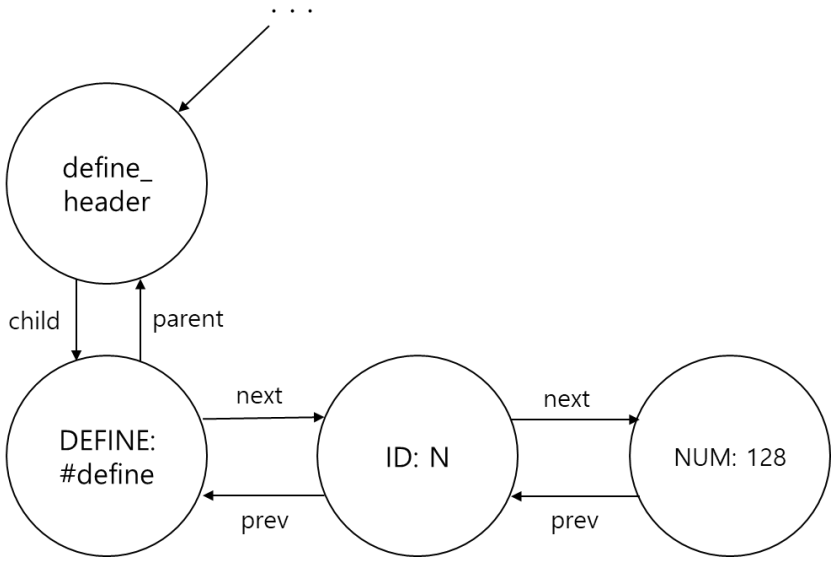


그림 4. Parse Tree for non-terminal "define_header"

Context-free-grammar 상의 non-terminal 과 terminal 이 parse tree 의 node 가 된다. 문법에 따라 non-terminal 을 derivation 할 때, action 을 통해 node 생성 및 parse tree 를 생성한다. Grammar 의 sentential form 에서 첫번째 non-terminal/terminal 이 child 가 되고, 이외의 non-terminal/terminal 들은 sibling 이 된다.

Bison 은 bottom-up parsing 을 사용한다. 따라서, 그림 4 의 경우, define, id, 그리고 num 의 node 를 생성하여 sibling 으로 연결한후 define_header node 를 생성하여 parent-child 관계로 연결한다.

문제 1에서 작성한 tree walk 알고리즘을 통해 build한 parse tree를 출력한다. 출력 방법은 다음과 같다.

- Non-terminal/terminal에 대한 derivation이 시작되면 (를 연다
- 해당 non-terminal/terminal에 대한 derivation이 끝나면)를 닫는다

제공된 output.txt를 참고하여, 동일한 출력이 나오도록 코드를 작성하고, 다음을 제출한다.

① (70 점) Parse tree를 build하고, 출력하는 코드가 포함된 project3.y와 project3.l

② (50 점) project3.y의 구현에 대한 설명이 포함된 보고서 project3.pdf

※ 제공된 코드를 수정했을 경우, 수정한 이유를 반드시 보고서에 포함시킨다

③ (30 점) Parse tree를 출력하여 output.txt와 동일한 출력이 나온 사진 project3.jpg