HW3 보고서

1.구현 자료구조 및 설명

1.1 전역변수

```
vector<string> input;
stack<string> st;
vector<int> error_msg;
vector<struct defun> defun_list;
```

이번 과제에서 필요한 defun list를 저장할 자료구조를 하나 추가하였다.

1.2 구조체

```
typedef struct defun {
  string keyword;
  vector<string> param;
  vector<string> expression;
}defun;

typedef struct TreeNode {
  string node;
  TreeNode* LNode;
  TreeNode* RNode;
}TreeNode;
```

Interprete를 할 때 필요한 tree 구조체는 동일하다.

DEFUN을 정의할 때, defun 구조체에 저장이 된다.

defun에서 keyword는 해당 함수의 이름을 저장하고, param은 입력받은 매개변수들을 나누어 vector에 저장한다. 또한 수식을 저장하는 expreesion이 존재한다.

1.3 함수

1.3.0 `int main() {}`

맨 처음 실행되는 main 함수이다. while문으로 무한 루프를 돌려 command를 입력받을 수 있도록 하였다.

1.3.1 `void command() {}`

사용자가 맨 처음 명령을 입력받을 때 필요한 함수이다.

switch-case문으로 1을 입력하였을 때 defineDefun()을, 2를 입력하였을 때 printDefun()을, 3을 입력하였을 때 interpreter()를, 4를 입력하였을 때 exit(1)을 할

수 있도록 하였다.

1.3.3 `void defineDefun() {}`

DEFUN 을 정의하는 함수이다.
newDefun이라는 Defun 구조체를 하나 만들어 이 구조체를 defun_list에 push_back한다. 사용자에게 입력 받은 input이 DEFUN ADD (x y) (MINUS x (MINUS 0 y)) 라면 ADD는 newDefun→keyword에, x,y는 param에, (MINUS x (MINUS 0 y))는 parse()함수를 통해 문자열을 전부 나누어 newDefun→expression에 저장한다.

1.3.3 `void printDefun() {}`

전역변수 defun_list에 저장되어 있는 defun들을 모두 출력해준다.

1.3.2 `void interpeter() {}`

사용자가 직접 문장을 입력할 경우이다. getline()으로 한 줄을 입력 받으며, 이것은 char line[]에 저장하여, operate함수로 넘겨준다.

1.3.4 `void operate(char line[]) {}`

operate()함수는 크게 parse(), func(), addNode(), postorder()로 구성되어있다. 네 함수를 실행시키고, 최종적으로 결과값을 출력해낸다.

error메세지를 출력하는 부분도 구성되어있다. error_check_num()이나 error_check_str()함수에서 에러가 발견되면, 해당 error메세지를 출력해주고 계산 없이 리턴한다.

1.3.5 `void parse(char line[]) {}`

line[]으로 입력줄을 받았으면, 이것을 괄호, 띄어쓰기로 파싱하여 전역변수인 input vector에 넣어준다.

예를들어, 사용자가 (IF 3 1)을 입력하였다면, input vector에 차례대로 (, IF, 3, 1,) 이 string 타입으로 들어가게 된다.

1.3.5 `int func() {}`

input 배열에 IF, MINUS, (,), 숫자 제외 다른게 있는지 확인하고, 다른게 존재하면 substitute() 함수를 호출하여 치환을 한다. 중첩 defun이 있을 수 도 있는 것을 고려하여, operate 함수에서 do-while문으로 전부 검사를 한다.

이 함수에서는 flag 변수가 존재한다. input 배열을 검사하여 IF, MINUS, (,), 숫자 만 존재한다면 0을, 그 외에 다른 문자열이 존재한다면 1을, 하지만 defun list에서 찾을 수 없을 경우에는 777을 리턴할 수 있도록 해주는 flag이다.

substitute()함수를 호출할 때, param_list와 keyIdx를 인자로 넘겨준다. param_list는 input 배열에서 파라미터에 해당하는 문자열의 시작, 끝 위치를 나타내는 배열이다.

1.3.5 `vector<string> substitute(vector<pair<int, int>> param_list, int idx) {}`

input 배열에 defun에 있는 함수명이 있으면 그 함수에 해당하는 수식으로 변환해주는 함수이다. 이 함수의 작동 과정은 다음과 같다.

ADD를 예로 설명하겠다. 우리가 바꿔주어야 할 input이 (ADD 1 2) 라고 가정하자.

1. idx로 어떤 defun list에서 ADD를 찾는다.
2. ADD의 파라미터 x를 cur 변수에 저장한다.
3. expression인 (MINUS x (MINUS 0 y)) tmp 변수에 복사한다.
4. tmp에서 cur의 위치를 찾고, tmp에서 그 위치에 해당하는 매개변수를 input 에 있는 매개변수와 바꿔준다.(x를 1로)

위 과정을 모든 param에 대해 바꿔주면, tmp배열은 IF와 MINUS만 존재하는 배열이 된다. 이 tmp를 리턴하여 func()에서 input의 (ADD 1 2)에 해당하는 부분을 (MINUS 1 (MINUS 0 2)) 로 바꿔준다. (아래 코드 참고)

```
for (int j = 0; j < tmp.size(); j++) {
      //defun struct의 param과 tmp[j] 위치가 같을때 (매개변수 위치 찾음)
      if (cur == tmp[j]) {
        int a = j;
        tmp.erase(tmp.begin() + j);
        //tmp[j] 위치에 input[param[i].first] 부터 input[param[i].second] 까지 삽입
        for (int k = param_list[i].first; k <= param_list[i].second; k++) {
          tmp.insert(tmp.begin() + a, input[k]);
          a++;
          j++;
        }

      }

    }
```

1.3.6 `TreeNode* addNode(int start, int end) {}`

파싱과 치환이 끝났다면, 최초에 addNode()가 operate()에서 불려졌을 때, 0과 input.size() -1 을 인자로 전달하여 실행이 된다. 이는 input 벡터의 첫번째 값, 마지막 값을 나타낸다.

이 함수는 재귀적으로 구성되어 있으며, 점차 start와 end의 범위를 좁혀 나간다. 재귀를 부르는 조건은 '(' 괄호의 갯수와 ')' 괄호의 갯수가 같을 때, 그 위치를 인자로 넣어주어 부른다. 또한, 왼쪽 오른쪽을 구분하여 넣는다.

함수가 실행될 때 마다 newNode 를 만들어 자신에게 값을 넣고, start와 end의 차이가 4라면 (예 : (IF 3 2)의 경우, start = 0, end = 4로 차이가 4이다.) 왼쪽, 오른쪽을 각각 삽입하도록 한다.

또한, 노드를 삽입할 때 마다 숫자와 문자열이 올바르게 들어가는지 error_check_str(), error_check_num() 함수로 검사한다.

1.3.7 `int postorder(TreeNode* ptr) {}`

addNode()로 만들어진 binary tree를 postorder로 읽어 스택에 저장한다. 재귀적으로 left를 먼저 방문해주고, right를 방문한다. stack에 저장시, MINUS나 IF를 만나면 pop()연산을 해 주어, calculate함수로 결과를 계산한다. 그리고 그 결과를 다시 스택에 push()해준다. 최종적으로, 스택에는 결과값만 남게 된다.

수식이 잘못되었을 경우를 대비하여, 잘못 되었을 때 에러메세지를 출력하고 777을 리턴하여 동작이 끝나도록 한다.

1.3.8 `string calculate(string op, string a, string b) {}`

MINUS일 경우, a-b를 리턴해준다. IF일 경우, a가 양수이면 b를 리턴하고, 그렇지 않으면 0을 리턴한다.

1.3.9 `void error_check_str(string check) {}`

문자열이 IF나 MINUS가 아니면 error_msg 벡터에 1을 push_back()한다.

1.3.10 `void error_check_num(string check) {}`

숫자가 실수이거나 -기호가 여러개 있을 경우, 해당하는 에러메세지 인덱스를 err_msg 벡터에 push_back()한다.

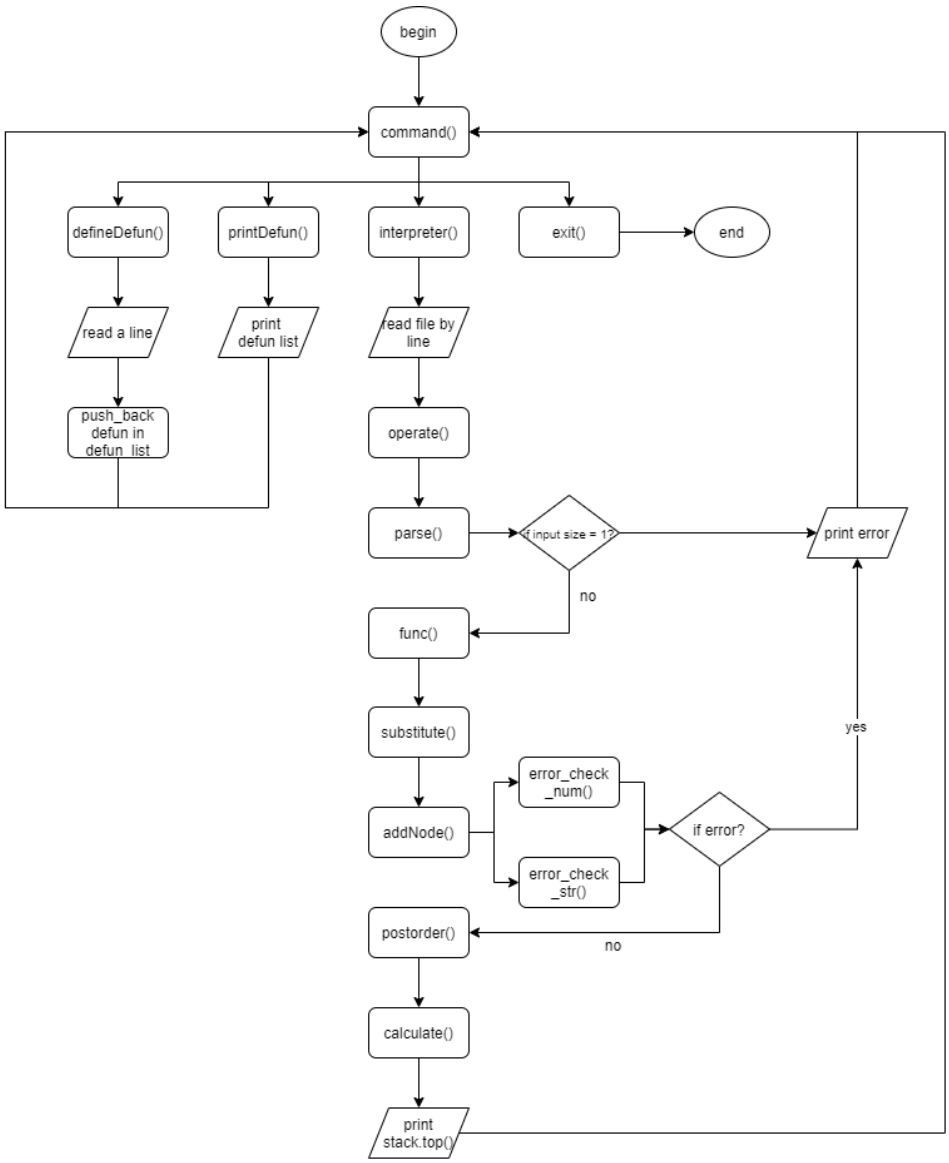
1.3.11 `bool isInt(string str) {}`

정수형인지 아닌지 검사하는 부가적인 함수이다. addNode에서 노드 삽입시 사용된다.

1.3.11 `int findKeyword(string str) {}`

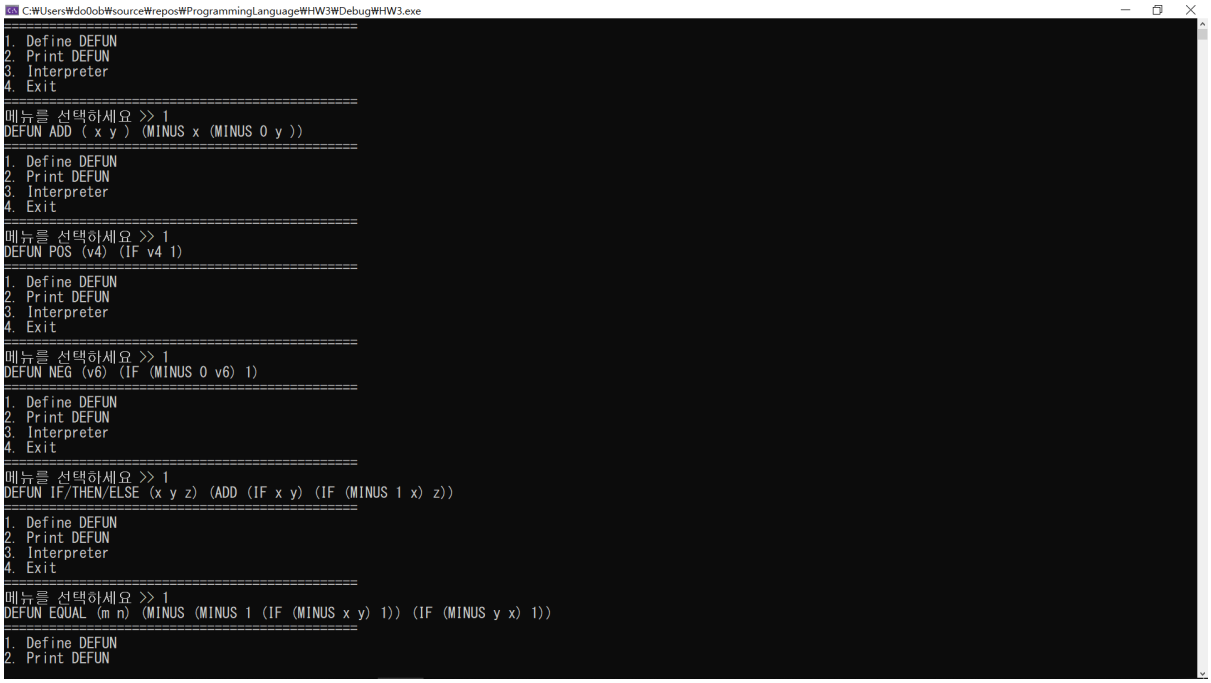
defun list에서 str에 해당하는 keyword가 존재하는지 검사한다. 있으면 그 위치를 반환, 없으면 777을 반환한다.

2.흐름도

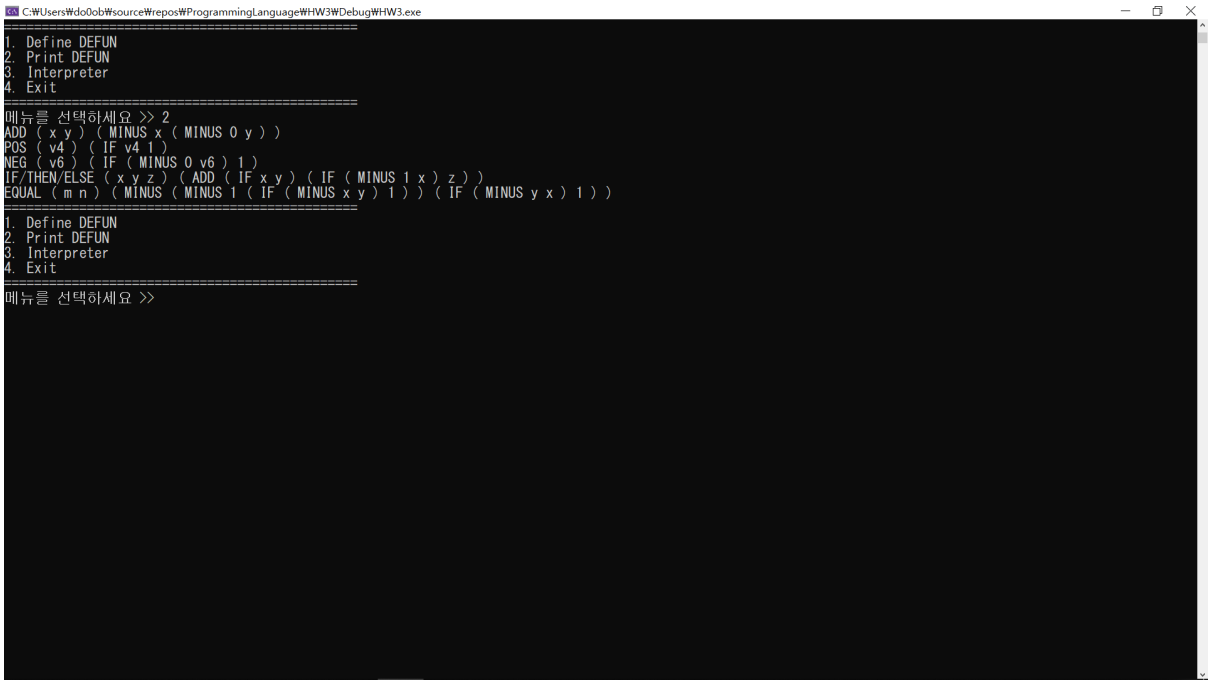

begin
command()
defineDefun()
printDefun()
interpreter()
exit()
end
read a line
print defun list
read file by line
push_back defun in defun_list
operate()
parse()
if input size = 1?
print error
no
func()
substitute()
yes
error_check _num()
addNode()
if error?
error_check _str()
postorder()
no
calculate()
print stack.top()

3.실행결과

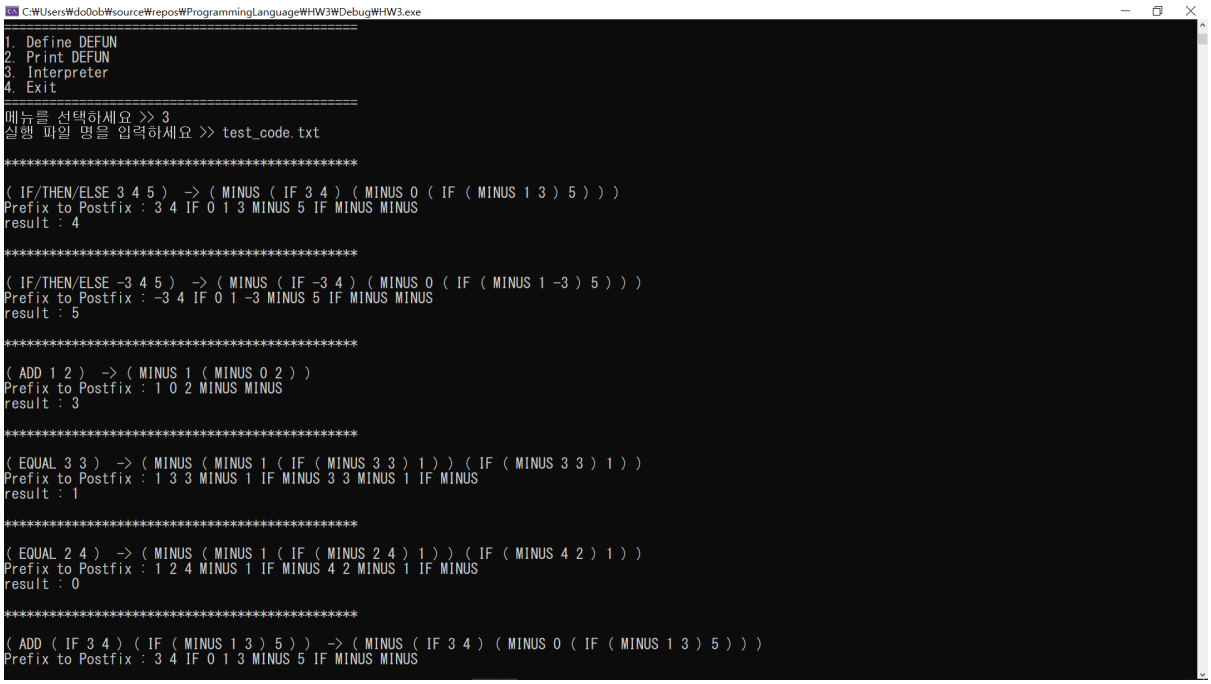
3.1 Define DEFUN



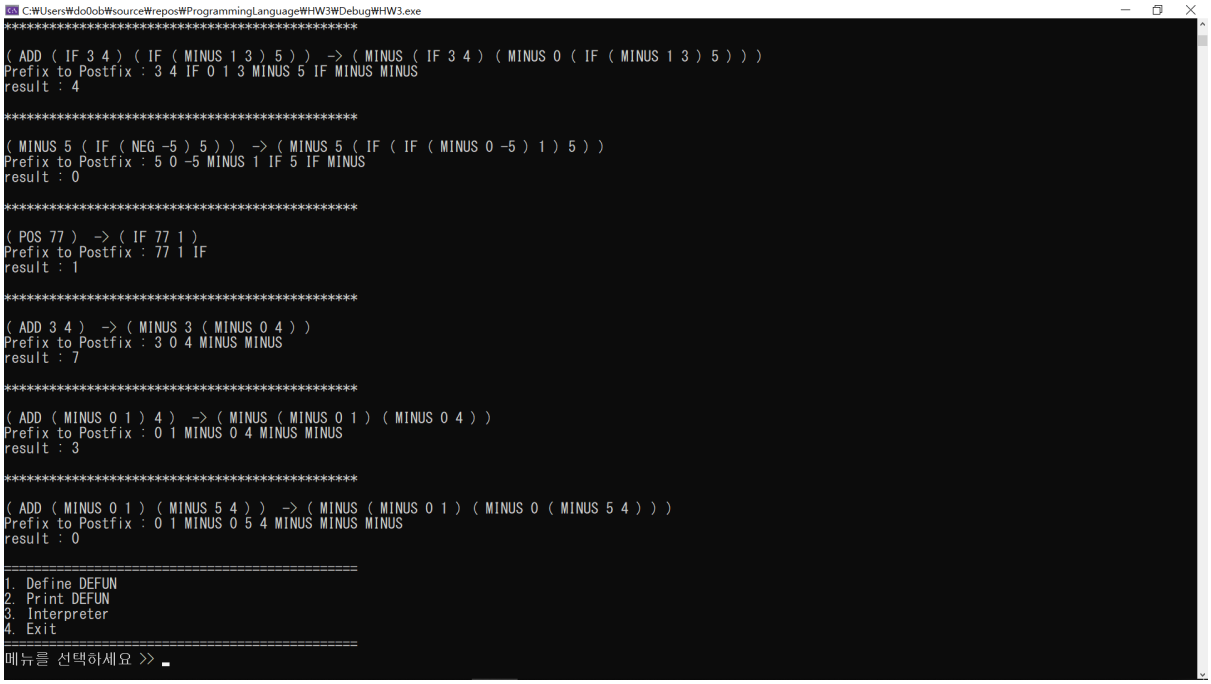
3.2 Print DEFUN



3.3 Interpreter - test_code.txt

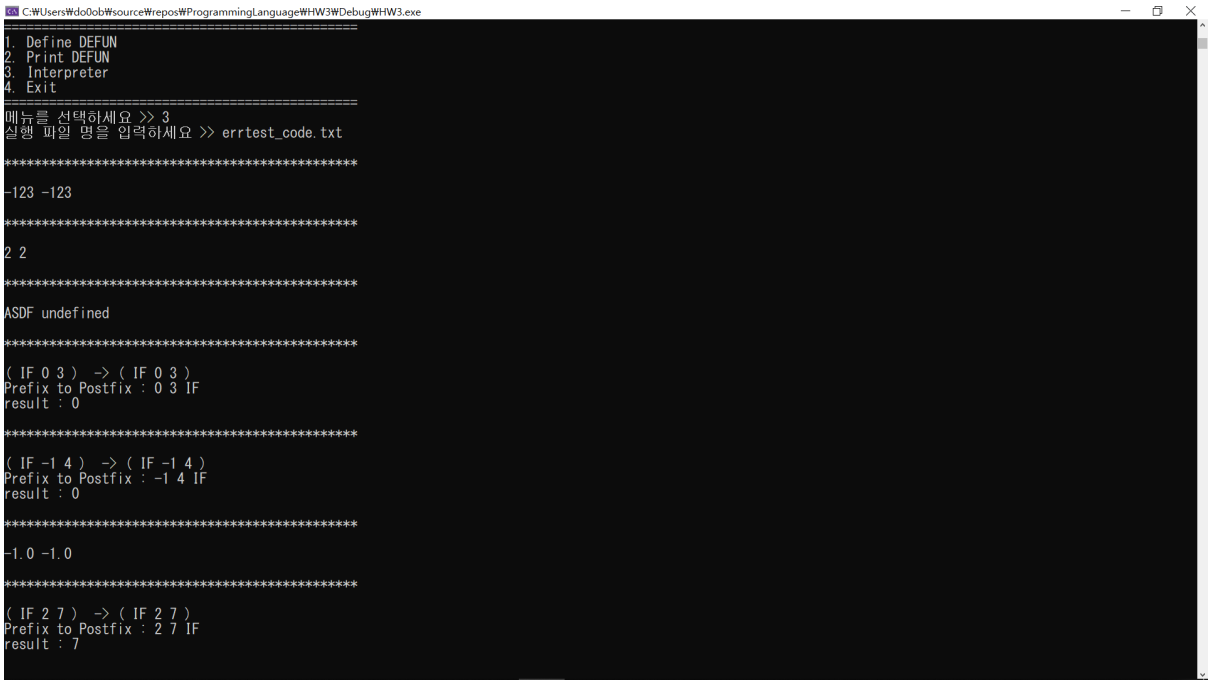


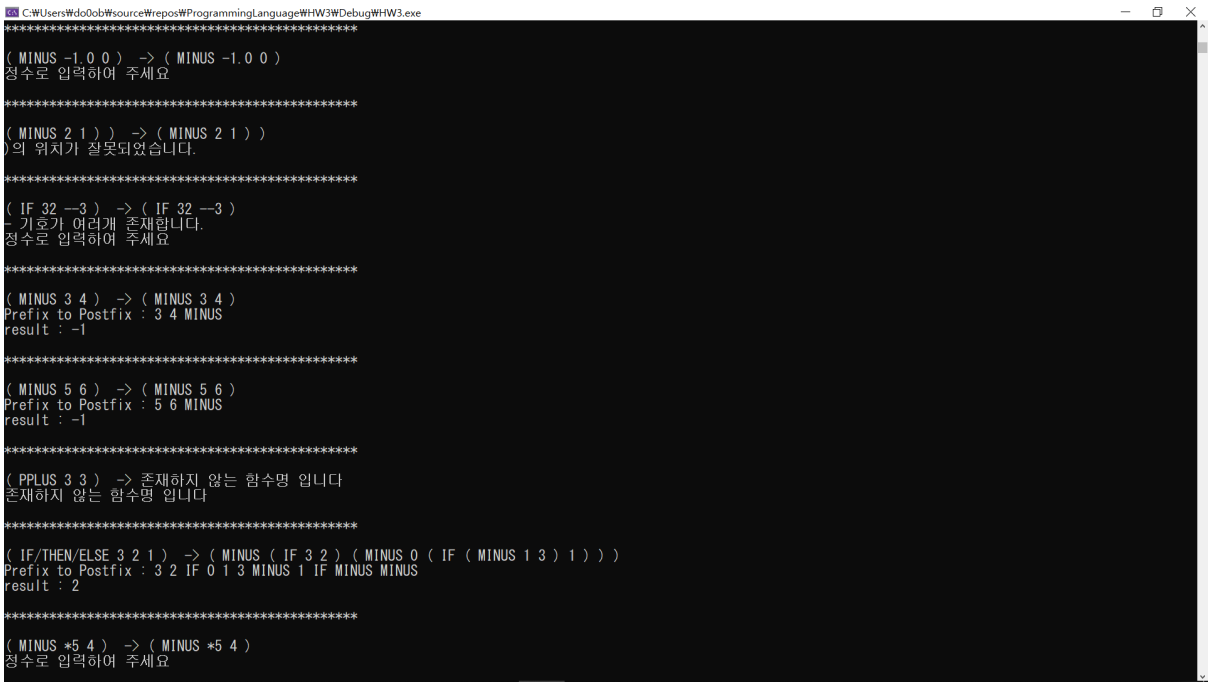
```
C:₩Users₩do0ob₩source₩repos₩ProgrammingLanguage₩HW3₩Debug₩HW3.exe
============================================
1. Define DEFUN
2. Print DEFUN
3. Interpreter
4. Exit
============================================
메뉴를 선택하세요 >> 3
실행 파일 명을 입력하세요 >> test_code.txt

************************************************

( IF/THEN/ELSE 3 4 5 )  -> ( MINUS ( IF 3 4 ) ( MINUS 0 ( IF ( MINUS 1 3 ) 5 ) ) )
Prefix to Postfix : 3 4 IF 0 1 3 MINUS 5 IF MINUS MINUS
result : 4

************************************************

( IF/THEN/ELSE -3 4 5 )  -> ( MINUS ( IF -3 4 ) ( MINUS 0 ( IF ( MINUS 1 -3 ) 5 ) ) )
Prefix to Postfix : -3 4 IF 0 1 -3 MINUS 5 IF MINUS MINUS
result : 5

************************************************

( ADD 1 2 )  -> ( MINUS 1 ( MINUS 0 2 ) )
Prefix to Postfix : 1 0 2 MINUS MINUS
result : 3

************************************************

( EQUAL 3 3 )  -> ( MINUS ( MINUS 1 ( IF ( MINUS 3 3 ) 1 ) ) ( IF ( MINUS 3 3 ) 1 ) )
Prefix to Postfix : 1 3 3 MINUS 1 IF MINUS 3 3 MINUS 1 IF MINUS
result : 1

************************************************

( EQUAL 2 4 )  -> ( MINUS ( MINUS 1 ( IF ( MINUS 2 4 ) 1 ) ) ( IF ( MINUS 4 2 ) 1 ) )
Prefix to Postfix : 1 2 4 MINUS 1 IF MINUS 4 2 MINUS 1 IF MINUS
result : 0

************************************************

( ADD ( IF 3 4 ) ( IF ( MINUS 1 3 ) 5 ) )  -> ( MINUS ( IF 3 4 ) ( MINUS 0 ( IF ( MINUS 1 3 ) 5 ) ) )
Prefix to Postfix : 3 4 IF 0 1 3 MINUS 5 IF MINUS MINUS
```

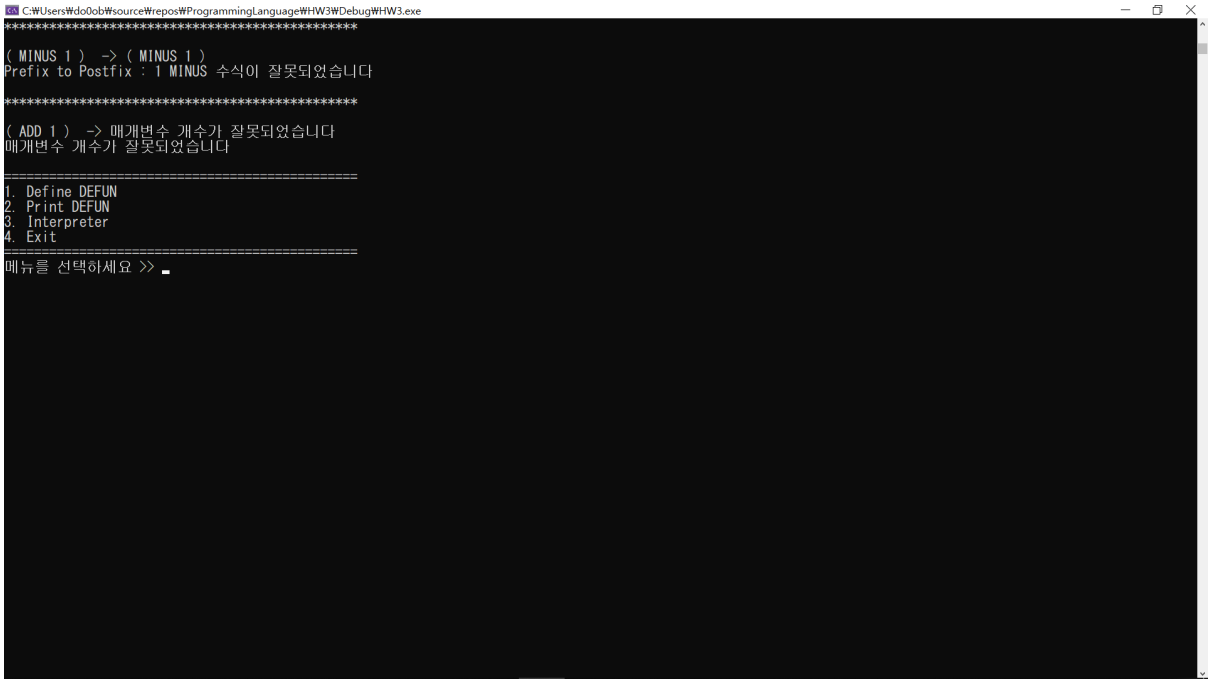


```
C:₩Users₩do0ob₩source₩repos₩ProgrammingLanguage₩HW3₩Debug₩HW3.exe
************************************************

( ADD ( IF 3 4 ) ( IF ( MINUS 1 3 ) 5 ) )  -> ( MINUS ( IF 3 4 ) ( MINUS 0 ( IF ( MINUS 1 3 ) 5 ) ) )
Prefix to Postfix : 3 4 IF 0 1 3 MINUS 5 IF MINUS MINUS
result : 4

************************************************

( MINUS 5 ( IF ( NEG -5 ) 5 ) )  -> ( MINUS 5 ( IF ( IF ( MINUS 0 -5 ) 1 ) 5 ) )
Prefix to Postfix : 5 0 -5 MINUS 1 IF 5 IF MINUS
result : 0

************************************************

( POS 77 )  -> ( IF 77 1 )
Prefix to Postfix : 77 1 IF
result : 1

************************************************

( ADD 3 4 )  -> ( MINUS 3 ( MINUS 0 4 ) )
Prefix to Postfix : 3 0 4 MINUS MINUS
result : 7

************************************************

( ADD ( MINUS 0 1 ) 4 )  -> ( MINUS ( MINUS 0 1 ) ( MINUS 0 4 ) )
Prefix to Postfix : 0 1 MINUS 0 4 MINUS MINUS
result : 3

************************************************

( ADD ( MINUS 0 1 ) ( MINUS 5 4 ) )  -> ( MINUS ( MINUS 0 1 ) ( MINUS 0 ( MINUS 5 4 ) ) )
Prefix to Postfix : 0 1 MINUS 0 5 4 MINUS MINUS MINUS
result : 0

============================================
1. Define DEFUN
2. Print DEFUN
3. Interpreter
4. Exit
============================================
메뉴를 선택하세요 >> _
```

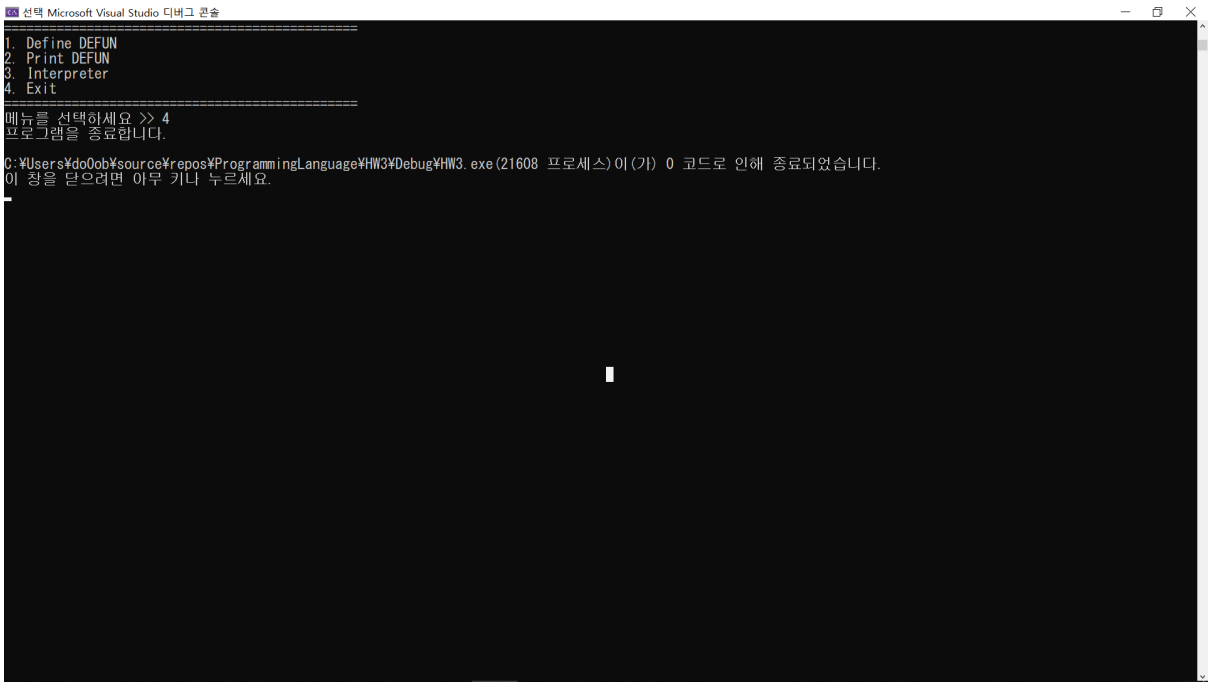
3.4 Interpreter - errtest_code.txt






C:₩Users₩do0ob₩source₩repos₩ProgrammingLanguage₩HW3₩Debug₩HW3.exe
****
(MINUS 1) -> (MINUS 1)
Prefix to Postfix : 1 MINUS 수식이 잘못되었습니다
****
(ADD 1) -> 매개변수 개수가 잘못되었습니다
매개변수 개수가 잘못되었습니다
==========
1. Define DEFUN
2. Print DEFUN
3. Interpreter
4. Exit
==========
메뉴를 선택하세요 >>

3.5 Exit


선택 Microsoft Visual Studio 디버그 콘솔
==========
1. Define DEFUN
2. Print DEFUN
3. Interpreter
4. Exit
==========
메뉴를 선택하세요 >> 4
프로그램을 종료합니다.
C:₩Users₩do0ob₩source₩repos₩ProgrammingLanguage₩HW3₩Debug₩HW3.exe(21608 프로세스)이(가) 0 코드로 인해 종료되었습니다.
이 창을 닫으려면 아무 키나 누르세요.