# 프로그래밍언어

# [과제3 TOY Language (version2)]


과목명 : 프로그래밍언어

교수명 : 양승민

학과 : 컴퓨터학부

학번 : 20160548

이름 : 이승현

## ● 구현 자료구조 및 함수 설명

### ➢ Main.cpp

void run(); // TOY언어 인터프리터를 실행하는 함수

void printMenu(); // 메뉴를 출력하는 함수

int inputNumber(); // 숫자를 입력받는 함수

char *getFileName(); // 파일명을 입력받는 함수

void printFileContents(FILE* fp); // 파일의 내용을 출력하는 함수

char* getTerm(); // 명령어를 입력받는 함수

char* loadTermsFromFile(); // 파일에서 명령어를 읽어오는 함수

bool executeFromFile(FILE* fp); // 파일의 명령어들을 실행시키는 함수

void executeInterpreter(); // 인터프리터를 실행시키는 함수

bool checkTerm(const char* term); // 함수에 매개변수가 하나 이상 있는지 확인하는 함수, 하나 이상 있으면 true 리턴, 새로운 함수를 정의한 문자열을 인자로 받는다

bool checkSameNameFunc(const char* term); // 이미 동일한 이름의 함수가 정의되어 있는지 확인하는 함수, 동일한 이름의 함수가 정의되어있다면 false 리턴, 새로운 함수를 정의한 문자열을 인자로 받는다

void appendNewTerm(const char* term); // 새로운 함수를 defun.txt에 추가하는 함수, 새로운 함수를 정의한 문자열을 인자로 받는다

void printAndLoadDefun(); // defun.txt에 정의되어 있는 사용자 정의 함수들을 콘솔창에 출력하여 사용자에게 보여주고, 해당 함수들을 프로세스에 load 하는 함수. 이 함수를 호출하여 load된 사용자 정의 함수들만 정상적으로 컴파일 할 수 있다.

struct termNode { // 명령어를 트리로 변환해서 저장한다. 그 트리의 노드 구조체

    const char* type; // 함수, 변수, 정수 세가지 타입이 있다

    termNode* param1; // 함수인 경우 첫번째 피연산자를 여기에 저장한다

    termNode* param2; // 함수인 경우 두번째 피연산자를 여기에 저장한다

    int value; // 정수인 경우 그 값을 여기에 저장

```
    termNode() :type(NULL), param1(NULL), param2(NULL), value(0) {} // 생성자


    void print() { // 디버깅을 위해 노드의 내용을 출력해보고 싶을 때 사용
        cout << "type : " << type << endl;
        cout << "param1 : " << param1 << endl;
        cout << "param2 : " << param2 << endl;
        if (!strcmp(INTEGER, type))
            cout << "value : " << value << endl;
    }
};
```

- **Compiler.cpp**

bool compile(char* terms); // 명령어를 컴파일하는 함수. 컴파일이란 명령어를 후위연산식으로 변환하고, 변환된 명령어를 텍스트 형식으로 파일에 출력해 중간 코드를 생성하는 것을 의미.

bool isWrongCharacter(char c); // 해당 문자가 TOY 언어에서 사용되면 안되는 문자인지 검사하여 사용하면 안되는 문자이면 true 리턴.

char* removeLeftWhitespace(char* terms); // 명령어 앞쪽의 공백을 제거하는 함수. 공백이 아닌 첫 번째 문자의 주소를 리턴한다.

void removeRightWhitespace(char* terms); // 명령어 뒤쪽의 공백을 제거하는 함수.

termNode* termToNode(char** terms); // 명령어를 트리로 변환하는 함수 재귀호출 함수.

bool checkTermsIncludingWrongCharacter(char* terms, char *wrongChar);

bool isInteger(char* term); // 전달받은 문자열이 정수 형식인지 확인하는 함수. 정수이면 true 리턴

void printAllTerms(termNode* node, FILE* fp); // 트리의 내용을 후위 연산식 형식으로 파일에 출력하는 함수

bool saveIntermediateCode(termNode* node); // 중간코드를 저장하는 함수

void freeAllNode(termNode* node); // 전달인자를 head로 하는 트리의 모든 노드를 free하는 함수

bool checkInvalidNumber(char* term); // 숫자가 입력되어야 할 자리에 "-"가 여러 번 연속으로 입력된 경우를 확인하는 함수. 숫자가 입력되어야 할 자리에 "-"가 여러 번 연속으로 입력되었으면 true 리턴

bool checkParenthesis(char* terms); // 괄호의 짝이 맞는지 확인하는 함수 괄호의 짝이 맞으면 true 리턴

string convertDescription(const char* term, const char* description); // 사용자 정의 함수를 TOY Language의 기본 함수로 이루어진 식으로 변환하는 함수, 명령어를 인자로 받아 사용자 정의 함수로 작성된 부분을 변환하고, 변환된 문자열을 리턴.

map<char*, char*> DEFUN; // load된 사용자 정의 함수들을 저장하는 map. 키는 함수의 이름이고, value는 함수의 정의 부.

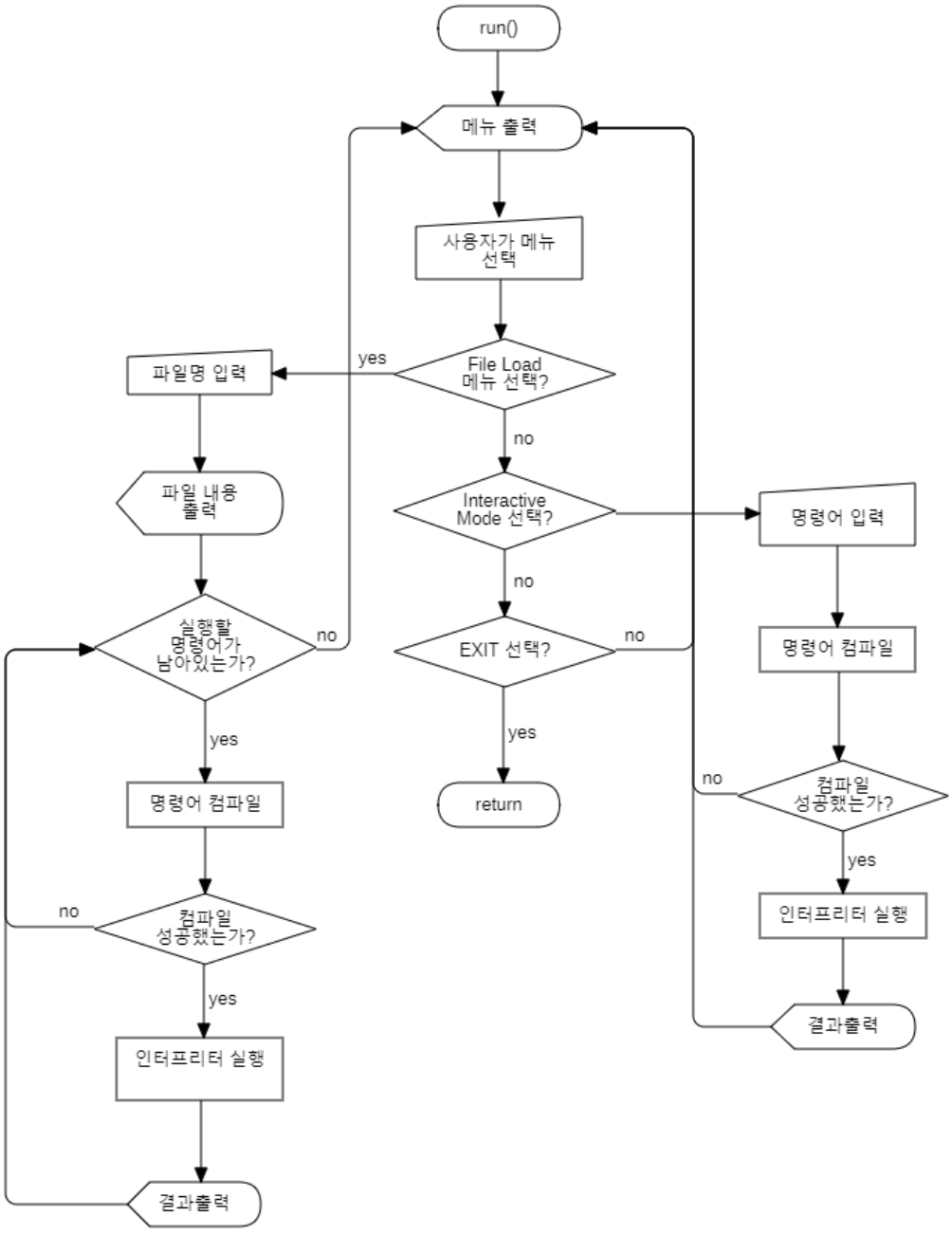
- **Interpreter.cpp**

void interpreter(char* terms); // 후위 연산식으로 변환된 명령어를 인자로 받아 실행하는 함수

## ● 주요 함수 흐름도

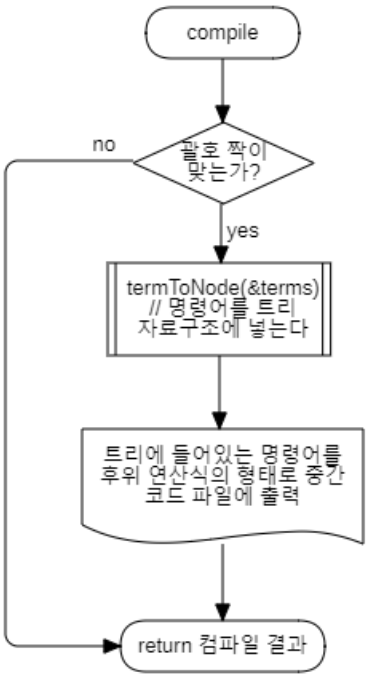
### ➢ Main.cpp

void run();

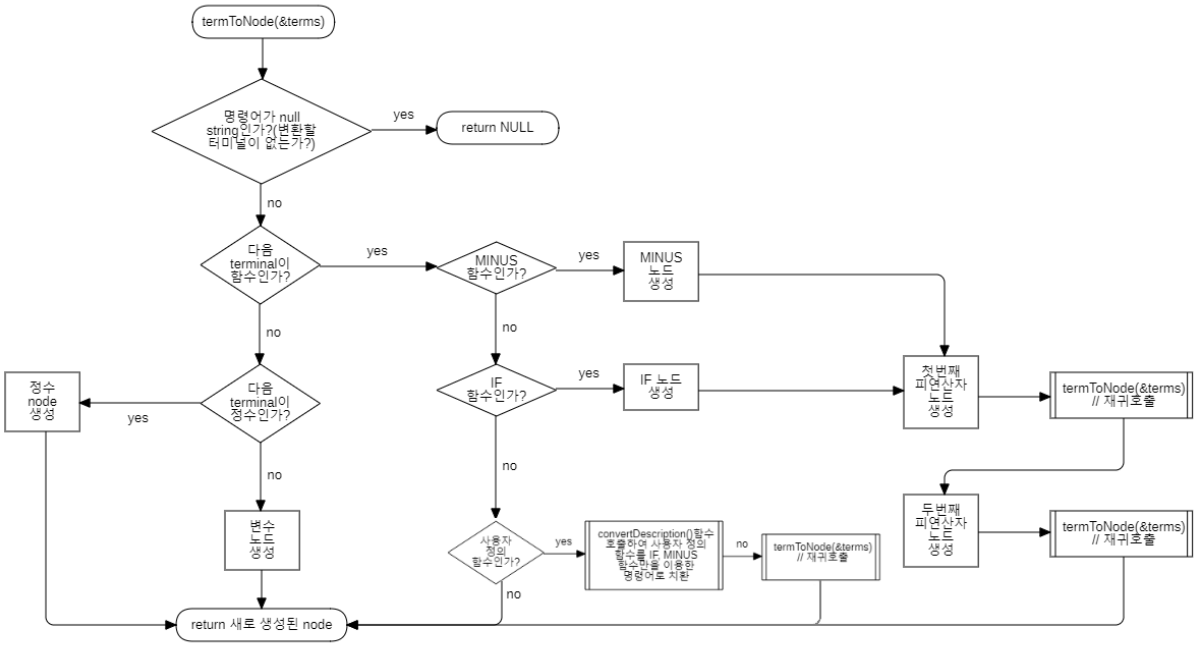


- **Compiler.cpp**

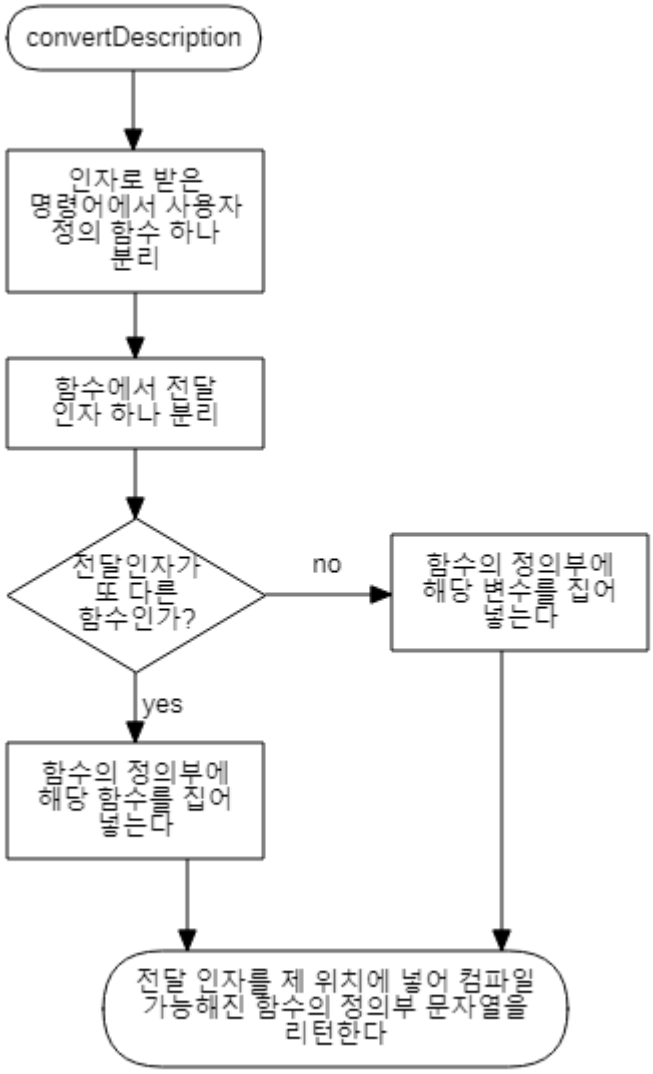
bool compile(char* terms);




termNode* termToNode(char** terms);

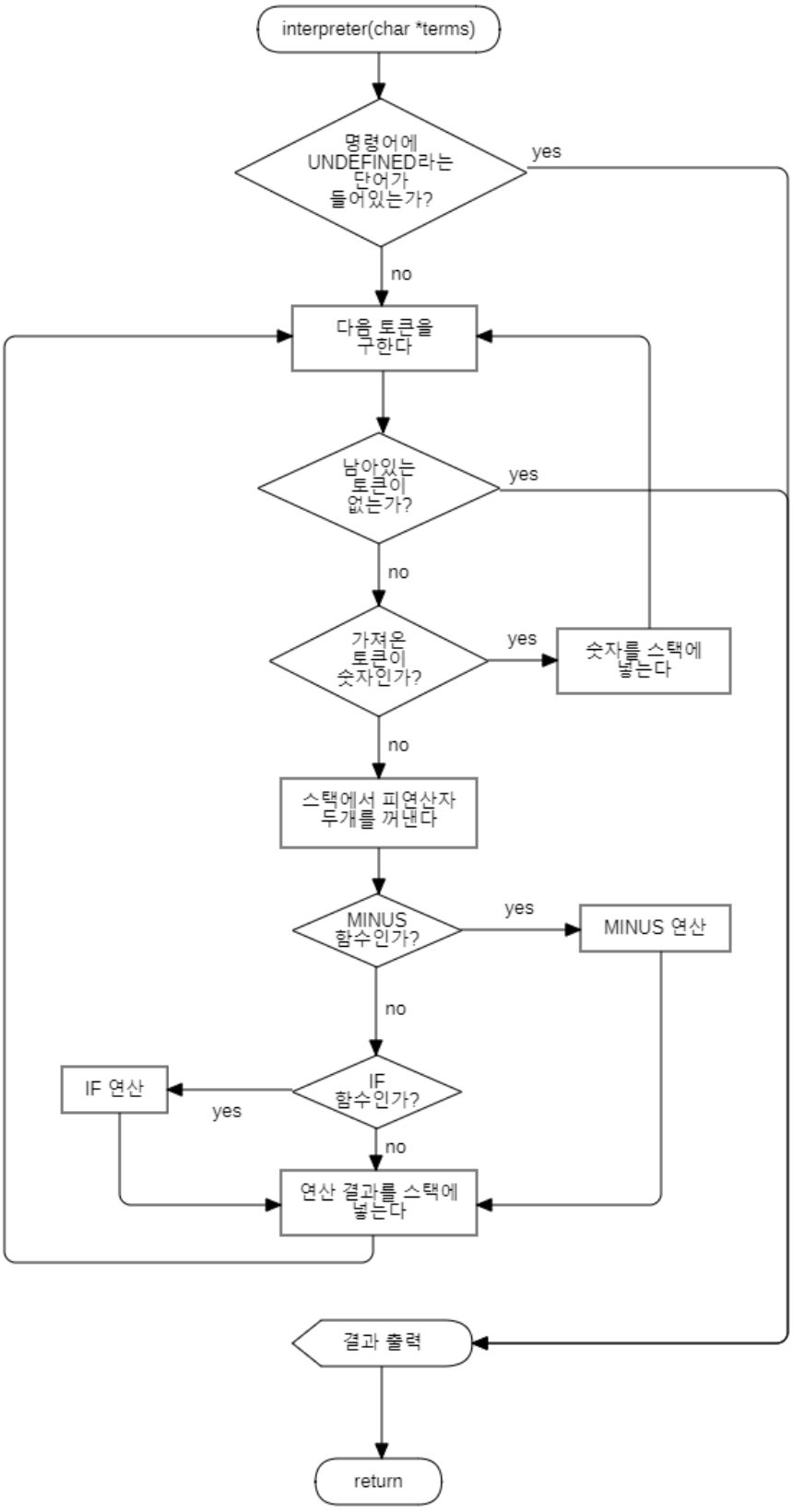


string convertDescription(const char *term, const char* description)



- **Interpreter.cpp**

void interpreter(char* terms);



## ● 실행 결과

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

DEFUN ADD (x y) (MINUS x (MINUS 0 y))
==================================
1. Define DEFUN
2. Print DEFUN
3. Interpreter
4. Exit
==================================
메뉴를 선택하세요 >> 2

ADD (x y) (MINUS x (MINUS 0 y))
==================================
1. Define DEFUN
2. Print DEFUN
3. Interpreter
4. Exit
==================================
메뉴를 선택하세요 >> 1

DEFUN POS (v4) (IF v4 1)
==================================
1. Define DEFUN
2. Print DEFUN
3. Interpreter
4. Exit
==================================
메뉴를 선택하세요 >> 1

DEFUN NEG (v6) (IF (MINUS 0 v6) 1)
==================================
1. Define DEFUN
2. Print DEFUN
3. Interpreter
4. Exit
==================================
메뉴를 선택하세요 >> 1

DEFUN IF/THEN/ELSE (v7 v8 v9) (ADD (IF v7 v8) (IF (MINUS 1 v7) v9))
==================================
1. Define DEFUN
2. Print DEFUN
3. Interpreter
4. Exit
==================================
메뉴를 선택하세요 >> 1

DEFUN TIMES (v10 v11) (IF/THEN/ELSE v10 (ADD v11 (TIMES (MINUS v10 1) v11)) 0)
```

⇨ 프로그램 실행 시 Define DEFUN, Load DEFUN, Interpreter, Exit의 4가지 메뉴를 선택할 수 있게 한다

⇨ Define DEFUN 선택 시, DEFUN을 사용하여 정의한 명령어를 defun.txt파일에 저장한다

⇨ 결과 값을 출력한 후 다시 메뉴를 선택한다.

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

DEFUN EQUAL (v2 v3) (MINUS (MINUS 1 (IF (MINUS v2 v3) 1)) (IF (MINUS v3 v2) 1))
==================================
1. Define DEFUN
2. Print DEFUN
3. Interpreter
4. Exit
==================================
메뉴를 선택하세요 >> 2

ADD (x y) (MINUS x (MINUS 0 y))
POS (v4) (IF v4 1)
NEG (v6) (IF (MINUS 0 v6) 1)
IF/THEN/ELSE (v7 v8 v9) (ADD (IF v7 v8) (IF (MINUS 1 v7) v9))
TIMES (v10 v11) (IF/THEN/ELSE v10 (ADD v11 (TIMES (MINUS v10 1) v11)) 0)
EQUAL (v2 v3) (MINUS (MINUS 1 (IF (MINUS v2 v3) 1)) (IF (MINUS v3 v2) 1))
```

⇨ Load DEFUN 선택 시, defun.txt에 정의되어 있는 DEFUN 명령어를 화면에 출력하고, 명령어를 load한다. 파일로 불러온 DEFUN값은 프로그램 끝날 때까지 유지된다

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

파일명을 입력하세요 >> test_code.txt

**********
(MINUS 1 (MINUS 0 2))

Prefix To Postfix : 1 0 2 MINUS MINUS
Result : 3
**********
(ADD 1 2)

Prefix To Postfix : 1 0 2 MINUS MINUS
Result : 3
**********
(POS 3)

Prefix To Postfix : 3 1 IF
Result : 1
**********
(POS -3)

Prefix To Postfix : -3 1 IF
Result : 0
**********
(MINUS 5 (IF (NEG -5) 5))

Prefix To Postfix : 5 0 -5 MINUS 1 IF 5 IF MINUS
Result : 0
**********
(NEG 3)

Prefix To Postfix : 0 3 MINUS 1 IF
Result : 0
**********
(NEG -7)

Prefix To Postfix : 0 -7 MINUS 1 IF
Result : 1
**********
(EQUAL 33 33)

Prefix To Postfix : 1 33 33 MINUS 1 IF MINUS 33 33 MINUS 1 IF MINUS
Result : 1
**********
(EQUAL 1 -1)

Prefix To Postfix : 1 1 -1 MINUS 1 IF MINUS -1 1 MINUS 1 IF MINUS
Result : 0
```

⇨ Interpreter 선택 시, 같은 폴더 내에 존재하는 프로그램 파일명(.txt파일)을 입력 받아 내용을 출력한 후 인터프리터를 실행한다. 인터프리터 실행이 끝나면 결과 값을 출력한다.

⇨ 복잡한 문장의 해석이 가능하도록 구현

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

파일명을 입력하세요 >> errtest_code.txt

**********
-123

Prefix To Postfix : -123
Result : -123
**********
2

Prefix To Postfix : 2
Result : 2
**********
ASDF

Prefix To Postfix : UNDEFINED
Result : undefined
**********
(IF 0 3)

Prefix To Postfix : 0 3 IF
Result : 0
**********
(IF -1 4)

Prefix To Postfix : -1 4 IF
Result : 0
**********
-1.0

"."는 사용할 수 없는 문자입니다. - 컴파일 에러
**********
(IF 2 7)

Prefix To Postfix : 2 7 IF
Result : 7
**********
(MINUS -1.0 0)

"."는 사용할 수 없는 문자입니다. - 컴파일 에러
**********
(MINUS 2 1))

")" 의 위치가 잘못되었습니다. - 컴파일 에러
**********
(IF 32 --3)

숫자가 입력되어야 할 자리에 "-"가 여러번 연속으로 입력되었습니다 - 컴파일 에러
**********
(MINUS 3 4)

Prefix To Postfix : 3 4 MINUS
Result : -1
**********
```

⇨ 문법에 맞지 않는 문장이 입력되면 어디가 잘못되었는지 체크하여 출력하도록 구현

```
**********
(PPLUS 3 3)

정의되지 않은 함수입니다. - 컴파일 에러
**********
(IF/THEN/ELSE 3 2 1)

Prefix To Postfix : 3 2 IF 0 1 3 MINUS 1 IF MINUS MINUS
Result : 2
**********
(MINUS *5 4)

"*"는 사용할 수 없는 문자입니다. - 컴파일 에러
**********
(MINUS 1)

Prefix To Postfix : 1 UNDEFINED MINUS
Result : undefined

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

DEFUN ERRTEST ()
매개변수가 하나 이상 있어야 합니다.
```

⇨ DEFUN으로 함수를 정의할 때 매개변수가 하나도 없을 시 에러 처리

```
==================================
메뉴를 선택하세요 >> 2

ADD (x y) (MINUS x (MINUS 0 y))
POS (v4) (IF v4 1)
NEG (v6) (IF (MINUS 0 v6) 1)
IF/THEN/ELSE (v7 v8 v9) (ADD (IF v7 v8) (IF (MINUS 1 v7) v9))
TIMES (v10 v11) (IF/THEN/ELSE v10 (ADD v11 (TIMES (MINUS v10 1) v11)) 0)
EQUAL (v2 v3) (MINUS (MINUS 1 (IF (MINUS v2 v3) 1)) (IF (MINUS v3 v2) 1))
==================================
1. Define DEFUN
2. Print DEFUN
3. Interpreter
4. Exit
==================================
메뉴를 선택하세요 >> 1

DEFUN ADD (samename errtest) ()
이미 정의된 함수입니다.

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

프로그램을 종료합니다.
```

⇨ DEFUN으로 함수를 정의할 때 이미 정의되어 있는 경우 에러 처리

⇨ Exit를 선택 시 프로그램을 종료한다.