-9주차 과제 보고서-

(item2)

(개인)

201404377 진승언

<문제 해결 방안>

```
static HashMap<String, Node> map = new HashMap<String, Node>(); // 추가
```

1. 테이블 값을 저장할 전역변수를 HashMap으로 만들어 주었다. 어떠한 Id에 값을 저장하는 것이므로 key와 value로 이루어진 HashMap이 가장 적절하다고 생각했다.

```
--------
public Node runExpr(Node rootExpr) { // 초기 노드 파싱
    if (rootExpr == null)
        return null;
    if (rootExpr instanceof IdNode) {
        ////////////////////////////////////////////////////// (item2) 디파인한것을 출력해줄 ex)
        ////////////////////////////////////////////////////// (define a 3) 이라했으면 a를 입력하면
        ////////////////////////////////////////////////////// a가아니라 3이나오게함
        if (lookupTable(rootExpr.toString()) != null) { // 테이블에 해당 정의가 있을 경우
            return lookupTable(rootExpr.toString());
        }
        //////////////////////////////////////////////////////
        else { // 그냥 Id반환
            return rootExpr;
        }
    } else if (rootExpr instanceof IntNode)
        return rootExpr;

    else if (rootExpr instanceof BooleanNode)
        return rootExpr;
    else if (rootExpr instanceof ListNode)
        return runList((ListNode) rootExpr); // 리스트인경우 Node runList(ListNode list)로 가짐
    else
        errorLog("run Expr error");
    return null;
```

2. runExpr(Node rootExpr)

=> IdNode타입이 매개변수로 왔을 때 define된 것인지 lookupTable을 이용해서 확인한다. (IdNode인지는 instanceof를 이용해서 확인이 가능하고 정의된 값인지는 lookupTable메소드를 호출해서 안다. 이 메소드는 값이 있으면 값을 반환하고 없으면 null을 반환시키는 것을 이용해 알 수 있다.(뒤에서도 계속 이것을 이용한다.) 만약 정의된 변수라면 테이블 값을 반환하고 아니라면 단순 id값을 반환시켜주었다.

```
--------
void insertTable(String id, Node value) { // 테이블 함수추가
    try {
        if (value instanceof ListNode) { // ( define a ( < 1 2 ) ) 같이 바로 리스트가 오고 이것은 계산되서 #T로 저장되야함
            value = runList((ListNode) value); // 리스트의 결과값을 받아옴(특수문자 없으면 그냥반환)
        }

        if (value instanceof ListNode) {    //각각의 형에 맞게 값을 테이블에 추가해줌
            map.put(id, (IntNode) value);
            System.out.println("..ListNode값이 테이블에 추가되었습니다");
        } else if (value instanceof QuoteNode) {
            map.put(id, (QuoteNode) value);
            System.out.println("..QuoteNode값이 테이블에 추가되었습니다");
        } else if (value instanceof IntNode) {
            map.put(id, (IntNode) value);
            System.out.println("..IntNode값이 테이블에 추가되었습니다");
        } else if (value instanceof BooleanNode) {
            map.put(id, (BooleanNode) value);
            System.out.println("..BooleanNode값이 테이블에 추가되었습니다");
        }else if(value instanceof IdNode) {
            map.put(id, (IdNode) value);
            System.out.println("..IdNode값이 테이블에 추가되었습니다");
        }
    } catch (Exception e) {
        errorLog("run Expr error");
    }

}
```

3. insertTable(String id, Node value)

=> String과 Node 타입을 매개변수로 받아 각각의 타입에 맞게 HashMap에

데이터를 추가시켜주었다. 그리고 리스트노드일 경우 사칙연산 같은 리스트 내용일 수 있으므로 계산결과 값을 담을 수 있도록 초기에 runList 메소드를 해주었다. id는 항상 idNode타입일 것이므로 String형으로 해야 저장하고 꺼내오는데 편리할거라 생각해서 String형으로 하였다.

```
Node lookupTable(String id) { // 테이블에서 데이터 꺼내오는 함수 추가
    if (map.containsKey(id)) {
        Node node = map.get(id);
        return node; // (define a 3)하면 ( 3 ) 이렇게 괄호 붙어서 반환
    } else
        return null; // 없을 경우 null반환
}
```

4. lookupTable(String id)

=> 이 메소드는 HashMap으로부터 define된 변수의 값을 가져오는 역할을 한다. HashMap의 key타입으로 String형을 썼으므로 이 메소드도 String형을 매개변수로 받아와서 get메소드를 이용해서 저장된 값을 반환 하게 하였다. 만약 저장되있지 않는 값이면 null을 반환시켰다.

```
case DEFINE:
    Node id_node = operand.car(); // 정의할 단어
    String id = id_node.toString();
    Node list = operand.cdr().car(); // 정의될 단어의 내용
    insertTable(id, list); // 테이블에 추가
```

5. define

=> 정의할 단어와 정의할 단어의 내용을 받아서 insertTable 메소드의 인자로 넘겨주었다. list의 경우 괄호 없이 저장 될 수 있게 cdr().car()을 해주었다.

```
switch (operator.value) { // CAR, CDR, CONS등에 대한 동작 구현
case CAR:
    ////////////////////////////////////////////////////////////////////////////////////////// (item2)

    if (operand.car() instanceof IdNode && lookupTable(((IdNode) operand.car()).toString()) != null ) { // IdNode일 때는 define한것 일 수도 있으므로 검사해준다
        Node result = ((QuoteNode) lookupTable(((IdNode) operand.car()).toString())).nodeInside(); // 다음 리스트반환
                                                                                                    // 괄호없이
                                                                                                    // 반환되므로
                                                                                                    // runQuote사용불가
        return ((ListNode) result).car(); // 그 리스트의 첫번째 값 반환
    }
    ////////////////////////////////////////////////////////////////////////

    else if (operand.car() instanceof QuoteNode) { // 현재 car다음에 오는 노드가 '인지 검사
        ListNode result = (ListNode) runQuote(operand);
        return result.car();
    } else {
        errorLog("run Expr error");
    }
    return null;
```

6. car

=> 기존의 코드의 앞에다가 car 뒤에 오는 값이 IdNode이면서 테이블에 있는 값이라면 그 정의된 내용을 lookupTable 메소드를 이용해 받아와서

result에 저장한 후 반환시켜주었다. 예를들어 입력이 (define a ` ( 1 2 ) ) 였다면 테이블을 통해 얻어오는 값은 ` (1 2) 이고 `를 제외한 값을 얻어오기위해 nodeinside메소드를 통해 (1 2) 리스트를 받아와서 result에 저장 후, 이 리스트의 car을 해주어 반환해주었다.

```
case CDR:
    ////////////////////////////////////////////////// (item2)

    if (operand.car() instanceof IdNode) { // IdNode일 때는 define한것 일 수도 있으므로 검사해준다
        Node result = ((QuoteNode) lookupTable(((IdNode) operand.car()).toString())).nodeInside(); // `다음 리스트반환
                                                                                                   // 괄호없이
                                                                                                   // 반환되므로
                                                                                                   // runQuote사용불가
        return ((ListNode) result).cdr(); // 그 리스트의 첫번째 값 반환
    }
    ////////////////////////////////////////////////////////

    else if (operand.car() instanceof QuoteNode) {
        ListNode result = (ListNode) runQuote(operand);
        return result.cdr();
    } else {
        errorLog("run Expr error");
    }
    return null;
```

7. cdr

=> car와 동일한 구조로 해주고 반환할 때 cdr로 반환해주었다.

```
case CONS: // ex ` (234)
    Node tmp = operand.car();
    Node tmp2 = operand.cdr().car();
    if (operand.car() instanceof IdNode && lookupTable(((IdNode) tmp).toString()) != null) { // 첫 노드가 테이블값
        tmp = lookupTable(((IdNode) tmp).toString()); // `(234)
    }
    if (operand.cdr().car() instanceof IdNode && lookupTable(((IdNode) tmp2).toString()) != null) { // 두번째 노드가
                                                                                                    // 테이블값
        tmp2 = lookupTable(((IdNode) tmp2).toString());
    }
    if (tmp instanceof QuoteNode) { // 처음께 `인 경우 (리스트가 오는 경우)
        Node result = ListNode.cons(((QuoteNode) tmp).nodeInside(), (ListNode) ((QuoteNode) tmp2).nodeInside());
        return result;
    } else {
        Node result = ListNode.cons(tmp, (ListNode) ((QuoteNode) tmp2).nodeInside());
        return result;
    }
```

8. cons

=> head부분과 tail부분을 받아와서 tmp, tmp2 변수에 저장한다. (앞부분과 뒷부분을 받아온다) 그 앞부분과 뒷부분이 idNode이면서 테이블에 있는 값인지 확인한다. 만약 있다면 lookupTable을 이용해서 정의된 내용을 tmp, tmp2에 저장해 변수를 바꿔준다. 그다음 if else문으로 처음에 QuoteNode타입이 올 경우와 아닌 경우로 나눠서 ListNode클래스의 cons 메소드를 tmp와 tmp2를 합쳐준다. 이 때, 앞에서 테이블 값으로 변경되었을 경우는 그 값으로 합쳐지고 아니면 기존 값들이 합쳐지게 했다.

```
case NULL_Q:
    ////////////////////////////////////////////////////// // 추가(item2) define문인지
    ////////////////////////////////////////////////////// 확인
    if (operand.car() instanceof IdNode && lookupTable(((IdNode) operand.car()).toString()) != null) { // null_q는
                                                                                                        // 리스트가오는
                                                                                                        // 경우인데
                                                                                                        // 이아이디가
                                                                                                        // 오는경우는
                                                                                                        // define문이
                                                                                                        // 오는경우임
        Node temp = ((QuoteNode) lookupTable(((IdNode) operand.car()).toString())).nodeInside(); // 리스트를 가져옴
        if (((ListNode) temp).car() == null) {
            BooleanNode result = BooleanNode.TRUE_NODE;
            return result;
        } else {
            BooleanNode result = BooleanNode.FALSE_NODE;
            return result;
        }
    }
    ////////////////////////////////////////////////////////////////////////////
    else if (((ListNode) runQuote(operand)).car() == null) {
        BooleanNode result = BooleanNode.TRUE_NODE;
        return result;
    } else {
        BooleanNode result = BooleanNode.FALSE_NODE;
        return result;
    }
```

9. null?

=> 기존코드의 앞에다가 null? 다음에 오는 값의 타입이 IdNode타입이면서 테이블에 저장된 값인지 확인해준다. 정의된 값이 맞다면 그 내용을 lookupTable을 이용해서 불러와서 null인지 체크한 후 null이면 True, 아니면 False를 반환해주는 것을 추가해줬다.

```
case COND:
    try {
        ListNode listNode = operand;
        while (listNode != ListNode.EMPTYLIST) {
            ////////////////////////////////////////////////////////////////// (item2)
            BooleanNode node;
            if (runExpr(((ListNode) listNode.car()).car()) instanceof IdNode
                    && lookupTable(((IdNode) ((ListNode) listNode.car()).car()).toString()) != null) { // if문으로 #T #F로 정의된 테이블 값이 올 경우
                node = (BooleanNode) lookupTable(((IdNode) ((ListNode) listNode.car()).car()).toString());
            }

            ////////////////////////////////////////////////////////////////////////////

            else { // 일반적인 #T #F가 왔을 경우
                node = (BooleanNode) runExpr(((ListNode) listNode.car()).car());
            }

            if (node == (BooleanNode.TRUE_NODE)) {

                ////////////////////////////////////////////////////////////////// (item2)

                if (((ListNode) listNode.car()).cdr().car() instanceof IdNode && lookupTable(
                        ((IdNode) (((ListNode) listNode.car()).cdr().car())).toString()) != null) { // 테이블 값일 경우
                    Node result = lookupTable(((IdNode) (((ListNode) listNode.car()).cdr().car())).toString());
                    if (result instanceof QuoteNode) { // 정의문이 Quote로 되있는 경우
                        return result; // (수정해줬음)
                    } else { // 워트문이 아닐경우
                        return result; // 위와 합쳐도되나 혹시모르니 그대로냅둠(마무리)
                    }

                }
                ////////////////////////////////////////////////////////////////////////////
                else {
                    return ((ListNode) listNode.car()).cdr().car();

                ////////////////////////////////////////////////////////////////////////////////////////
                else {
                    return ((ListNode) listNode.car()).cdr().car();
                }
            } else {
                listNode = operand.cdr(); // 다음 리스트로 부른다
            }
        }
    } catch (Exception e) {
        errorLog("run Expr error");
    }
```

10. cond

=> cond는 다음에 ( ) 리스트가 오고 앞에 값은 if문 역할을 하고 true면 뒤에 값을 반환하고 false면 반한하지 않는다. 기존의 코드의 앞에다가 if문 역할을 하는 리스트의 앞에 값이 idNode타입이면서 정의된 값이라면 해당 값의 내용을 lookupTable을 이용해서 가져온다. 그리고 그 값이 true(#T)이면 그 뒤에 값을 출력해주는 것을 추가해줬다. 그리고 그 뒤에 값도 정의된 값일 수도 있으므로 검사해줘서 정의된 값이라면 정의된 값을 return해주고 아니라면 그냥 값을 return해주는 것을 추가해주었다.

```
case NOT:
    ////////////////////////////////////////////////////////////////////////////// (item2)
    if (operand.car() instanceof IdNode && lookupTable(((IdNode) operand.car()).toString()) != null) { // 다음으로 테이블 값이 올 경우
        Node temp = lookupTable(((IdNode) operand.car()).toString()); // define a #T 인 경우 ( #T ) 값가져옴]
        if (temp instanceof BooleanNode) {
            if (temp == BooleanNode.TRUE_NODE)
                return BooleanNode.FALSE_NODE;
            else
                return BooleanNode.TRUE_NODE;
        }
    }
    ////////////////////////////////////////////////////////////////////////////
    else {
        if (operand.car() instanceof BooleanNode) {
            if (operand.car() == BooleanNode.TRUE_NODE)
                return BooleanNode.FALSE_NODE;
            else
                return BooleanNode.TRUE_NODE;
        } else if (((ListNode) operand.car()).car() instanceof FunctionNode) {
            if (runFunction((FunctionNode) ((ListNode) operand.car()).car(),
                    ((ListNode) operand.car()).cdr()) == BooleanNode.TRUE_NODE)
                return BooleanNode.FALSE_NODE;
            else
                return BooleanNode.TRUE_NODE;
        } else if (((ListNode) operand.car()).car() instanceof BinaryOpNode) {
            if (runBinary((ListNode) operand.car()) == BooleanNode.TRUE_NODE)
                return BooleanNode.FALSE_NODE;
            else
                return BooleanNode.TRUE_NODE;
        } else {
            errorLog("run Expr error");
            return null;
        }
    }
```

11. not

=> not 다음에 오는 값이 #T나 #F로 정의된 id값일 수도 있으므로 그 경우를 추가해주었다. if문으로 정의된 값인지 확인하고 그 값을 받아와서 true면 false , false면 true를 반환시켜주는 것을 추가해줬다.

```
case ATOM_Q:
    ////////////////////////////////////////////////// (item2)
    if (operand.car() instanceof IdNode && lookupTable(((IdNode) operand.car()).toString()) != null) { // define은
                                                                                                        // 일 경우
        Node temp = ((QuoteNode) lookupTable(((IdNode) operand.car()).toString())).nodeInside(); // 값 추출
        if (temp instanceof IdNode || temp instanceof IntNode) { // 리스트 아님
            return BooleanNode.TRUE_NODE;
        } else if (temp instanceof ListNode) // 리스트임
            return BooleanNode.FALSE_NODE;


    } else {
        ////////////////////////////////////////////////

        if(operand.car() instanceof QuoteNode) { // `없이 단순 값이왔을때 ex) 3, A
        if (runQuote(operand) instanceof IdNode || runQuote(operand) instanceof IntNode) { // 리스트 아님
            return BooleanNode.TRUE_NODE;
        } else if ((ListNode) runQuote(operand) instanceof ListNode) // 리스트임
            return BooleanNode.FALSE_NODE;
        else {
            errorLog("run Expr error");
            return null;
        }
    }
    else {
        return BooleanNode.TRUE_NODE;
    }
    }
```

12. atom?

=> atom은 리스트이면 false 아니면 true를 반환한다. 지금까지 해왔던 것처럼 atom? 다음에 오는 것이 idNode타입이면서 정의된 값일 경우 그 값의 내용을 불러와서 temp에 저장하였다. 그 후 temp가 ListNode타입이면 false 아니라면 true를 반한하게 해주었다. 또한 `(quote) 없이 바로 3, A 이런게 온 경우도 리스트가 아니므로 True를 반환하게 하기위해 다음에 온 값이 QuoteNode가 아닐 때도 검사하게 해주었다.

```
case EQ_Q:
    if (operand.car() instanceof IdNode) { // 첫노드가 id일 때 (
        if (lookupTable(operand.car().toString()) != null) { // 첫 노드가 테이블값일 때
            if (((ListNode) operand.cdr()).car() instanceof IdNode) { // 두번째가 id일때
                if (lookupTable((((ListNode) operand.cdr()).car()).toString()) != null) { // 두번째가 테이블값일 때
                    if (lookupTable(operand.car().toString()) instanceof QuoteNode) { // 테이블값이 quote일때
                        if (lookupTable(((operand.cdr()).car()).toString()) instanceof QuoteNode) { // 뒤의 테이블값이
                                                                                                    // quote일 때
                            ListNode temp = (ListNode) ((QuoteNode) lookupTable(operand.car().toString()))
                                    .nodeInside();
                            ListNode temp2 = (ListNode) ((QuoteNode) lookupTable(
                                    (((ListNode) operand.cdr()).car()).toString())).nodeInside();

                            while (temp.car() != null && temp.cdr() != null && temp2.car() != null
                                    && temp2.cdr() != null) { // 리스트끼리의 값 비교
                                if (temp.car().toString().equals(temp2.car().toString())) {
                                    temp = temp.cdr();
                                    temp2 = temp2.cdr();
                                    continue;
                                } else
                                    return BooleanNode.FALSE_NODE; // 하나라도 다르면 FALSE
                            }
                            if (temp.car() != null || temp.cdr() != null || temp2.car() != null
                                    || temp2.cdr() != null)
                                return BooleanNode.FALSE_NODE; // 개수비교
                            else
                                return BooleanNode.TRUE_NODE;
                        } else { // 앞 테이블 값은 쿼트인데 뒤는 쿼트아닐때 false
                            return BooleanNode.FALSE_NODE;
                        }
                    } else { // 처음 테이블값이 쿼트가아니라 한개일때
                        if (((ListNode) operand.cdr()).car() instanceof ListNode) { // 뒤에거는 리스트일경우 false
                            return BooleanNode.FALSE_NODE;
```

```
                } else { // 처음 테이블값 두번째 테이블값 모두 단수일 때 (비교해줌)
                    if (lookupTable(operand.car().toString()).toString().equals(
                            lookupTable((((ListNode) operand.cdr()).car()).toString()).toString()))
                        return BooleanNode.TRUE_NODE; // 앞에 변수와 뒤에 변수에 대한 EQ_Q
                    else
                        return BooleanNode.FALSE_NODE;
                }
            }


            } else { // 두번째가 테이블값이 아니라 id 일 때
                if ((lookupTable(operand.car().toString())).toString()
                        .equals((operand.cdr().car()).toString()))
                    return BooleanNode.TRUE_NODE;
                else
                    return BooleanNode.FALSE_NODE;
            }
        } else { // 두번째 노드가 Int일때
            if ((lookupTable(operand.car().toString())).toString().equals((operand.cdr().car()).toString()))
                return BooleanNode.TRUE_NODE;
            else
                return BooleanNode.FALSE_NODE;
        }
    } else { // 첫번째 값이 테이블값이 아니라 id일때
        if (operand.car().toString().equals((operand.cdr().car()).toString()))
            return BooleanNode.TRUE_NODE;
        else
            return BooleanNode.FALSE_NODE;
    }
} else if (operand.car() instanceof IntNode) { // 첫번째 값이 Int일때
    if (((ListNode) operand.cdr()).car() instanceof IdNode) { // 두번째 값이 id일때
        if (lookupTable((((ListNode) operand.cdr()).car()).toString()) != null) {
            if (lookupTable((((ListNode) operand.cdr()).car()).toString()) instanceof ListNode) {
                return BooleanNode.FALSE_NODE;
            } else if (operand.car().toString()
                    .equals(lookupTable((((ListNode) operand.cdr()).car()).toString()).toString()))
                return BooleanNode.TRUE_NODE;
            else
                return BooleanNode.FALSE_NODE;
        }
    } else { // 두번째 값이 int일때
        if (operand.car().toString().equals((operand.cdr().car()).toString()))
            return BooleanNode.TRUE_NODE;
        else
            return BooleanNode.FALSE_NODE;
    }
} else if (runQuote(operand) instanceof IntNode || runQuote(operand) instanceof IdNode) { // Int,Id 일때
    if (runQuote(operand).toString().equals(runQuote(operand.cdr()).toString()))
        return BooleanNode.TRUE_NODE;
    else
        return BooleanNode.FALSE_NODE;
} else { // 리스트일때
    ListNode temp = (ListNode) runQuote((ListNode) operand);
    ListNode temp2 = (ListNode) runQuote((ListNode) operand.cdr());

    while (temp.car() != null && temp.cdr() != null && temp2.car() != null && temp2.cdr() != null) { // 리스트값들
                                                                                                     // 비교
        if (temp.car().toString().equals(temp2.car().toString())) {
            temp = temp.cdr();
            temp2 = temp2.cdr();
            continue;
        } else
            return BooleanNode.FALSE_NODE;
    }
    if (temp.car() != null || temp.cdr() != null || temp2.car() != null || temp2.cdr() != null)
        return BooleanNode.FALSE_NODE; // 리스트 개수비교
    else
        return BooleanNode.TRUE_NODE;
}
```

13. eq?

=> 기존에 했던 코드에서 각각의 경우마다 idNode가 오는 경우 정의된 값인지 확인하고 정의된 값일 경우 그 값으로 비교를 하게끔 코드를 추가시켜주었다. eq? 다음에 오는 첫 번 째 값이 정의된 값인지 확인하고 두 번째 값도 정의된 값인지 확인시켜주는 것을 추가시켜주었다.

---

14. 사칙연산 (BinaryOpNode)
=> 이 부분은 수정할 것이 없었다.

```
private void printNode(Node node) {
    if (node == null)
        return;
    if (node instanceof ListNode) {
        ps.print("(");
        printNode((ListNode) node);
        ps.print(" )");
    } else if (node instanceof QuoteNode) {
        printNode((QuoteNode) node);
    } else if (node instanceof BooleanNode) {
        if (node == BooleanNode.FALSE_NODE)
            ps.print("#F");
        else
            ps.print("#T");
    }
    ////////////////////(item2 추가)
    else if (node instanceof FunctionNode) {
        ps.print((node.toString().toLowerCase()));
    }
    /////////////////
    else {
        String temp = node.toString();
        StringTokenizer st = new StringTokenizer(temp, " ");
```

15. NodePrinter클래스 부분 수정
=> 포털 사이트에 올라온 (car ' (cdr ' (1 2 3))) 예시의 이런 car cdr 끼리 중첩이 되었을 때를 해결하기위해 NodePrinter클래스의 printNode 메소드 부분을 수정하였다. car을 했을 때 cdr이 매개변수로 넘어오긴 하는데 기존코드에서 FunctionNode 값을 출력이 불가능하길래 다음 주석사이와 같이 출력이 되게끔 수정해주었다.

<실행 결과>

=>기존예시 (ppt파일에 있는 예시)

```
>( car ` ( 2 3 4 ) )
..  2
>( car ` ( ( 2 3 ) ( 4 5 ) 6 ) )
.. ( 2 3 )
>( cdr ` ( 2 3 4 ) )
.. ( 3 4 )
>( cdr ` ( ( 2 3 ) ( 4 5 ) 6 ) )
.. (( 4 5 ) 6 )
>( cdr ` ( 2 ) )
.. ( )
>( cons 1 ` ( 2 3 4 ) )
.. ( 1 2 3 4 )
```

```
>( cons ` ( 2 3 ) ` ( 4 5 6 ) )
.. (( 2 3 ) 4 5 6 )
> ( cons 2 ` ( ) )
.. ( 2 )
>( null? ` ( ) )
.. #T
>( null? ` ( 1 2 ) )
.. #F
>( null? ` ( ( ) ) )
.. #F
>( atom? ` a )
.. #T
>( atom? ` ( 1 2 ) )
.. #F
>( eq? ` a ` a )
.. #T
>( eq? ` a ` b )
.. #F
>( eq? ` ( a b ) ` ( a b ) )
.. #T
>( + 1 2 )
..  3
>( + ( + 1 2 ) 3 )
..  6
>( > 1 5 )
.. #F
>( > ( + 9 3 ) 5 )
.. #T
>( not #F )
.. #T
>( not ( < 1 2 ) )
.. #F
>( cond ( ( > 1 2 ) 0 ) ( #T 1 ) )
```

```
>( cond ( ( > 1 2 ) 0 ) ( #T 1 ) )
..  1
>( cond ( #F 3 ) ( #T 2 ) )
..  2
>( cond ( #T 2 ) ( #T 1 ) )
..  2
>
```

이 밑으로는 포털 사이트에 있는 예시이다

```
>( car ' ( cdr ' ( 1 2 3 ) ) )
.. cdr
>
```

```
> ( define a 1 )
..IntNode값이 테이블에 추가되었습니다
..
>( car ` ( a 2 3 ) )
..  a
>( define b ` ( 1 2 3 ) )
..QuoteNode값이 테이블에 추가되었습니다
..
>( car b )
..  1
>( define a 2 )
..IntNode값이 테이블에 추가되었습니다
..
>( define b 2 )
..IntNode값이 테이블에 추가되었습니다
..
>( eq? a b )
.. #T
>( define a 3 )
..IntNode값이 테이블에 추가되었습니다
..
>( define b 3 )
..IntNode값이 테이블에 추가되었습니다
..
>( eq? ` a ` b )
.. #F
>( define a ` ( 1 2 3 ) )
..QuoteNode값이 테이블에 추가되었습니다
..
>( define b ` ( 1 2 3 ) )
..QuoteNode값이 테이블에 추가되었습니다
```

```
>( define b ` ( 1 2 3 ) )
..QuoteNode값이 테이블에 추가되었습니다
..
>( eq? a b )
.. #T
>( define a ` ( 2 3 ) )
..QuoteNode값이 테이블에 추가되었습니다
..
>( cons 1 a )
.. ( 1 2 3 )
>( define a 2 )
..IntNode값이 테이블에 추가되었습니다
..
>( cond ( #T a ) ( #F 6 ) )
..  2
>( define a ` ( 2 3 ) )
..QuoteNode값이 테이블에 추가되었습니다
..
>( cond ( #T a ) ( #F 6 ) )
..  '( 2 3 )
>( define a #T )
..BooleanNode값이 테이블에 추가되었습니다
..
>( cond ( a ` ( 1 2 ) ) ( #F 6 ) )
..  '( 1 2 )
>( define a ( < 1 2 ) )
..BooleanNode값이 테이블에 추가되었습니다
..
>a
.. #T
> ( define a ` ( ) )
..QuoteNode값이 테이블에 추가되었습니다
..
```

```
.. #T
>
```

<느낀점>

=> 드디어 option을 제외한 과제가 끝났다. 기존예시를 제외한 내가 생각한 경우도 실행시켜보면서 최대한 오류를 없애기 위해 노력을 하였다. 물론 내가 생각한 것 외에 오류가 있을 수도 있겠지만 예시는 잘 돌아가서 다행이고 마무리가 돼서 기뻤다. 완성했다 싶으면 다른 경우의 수를 생각하여 다시 실행시켰을 때 에러가 나서 다시 고치는 것을 많이 반복하였다. 그래서 스트레스도 받고 괜히 다른 예시를 생각했다고도 느꼈지만 결과적으론 나름 더 괜찮은 인터프리터가 된 것에 만족을 하고 성취감을 느꼈다.
첫 주차 개미수열부터 시작해서 인터프리터를 차근차근 만들기 까지 고생을 많이 했고 배운점도 많은 것 같다. 이 인터프리터가 smalltalk언어 문법이라 들었는데 이 문법은 시간이 지나도 까먹지 않을 것 같다. 모든 과제를 java언어를 이용해서 수행하였는데 하면서 부족한 점도 많이 느꼈고 방학 때 java8 버전 공부와 알고리즘 공부 좀 해야겠다고 느꼈다. 마지막으로 질문에 답변을 친절히 해주신 조교님께 감사하다.

<구동방식>

=> `(quote)포함 모든 값들이 띄어쓰기된 상태로 입력해야 실행됩니다.