

PL Assignment #7: Cute15 Built-in Function 구현

과제물 부과일 : 2015-04-27 (월)

Program Upload 마감일 : 2015-05-03 (일) 23:59:59

문제

Cute15 문법에 따라 작성된 program이 as07.txt에 저장되어 있다. 이 프로그램은 Cute15의 built-in function을 사용하여 리스트를 조작하며 그 결과를 구하고 있다. 이러한 프로그램을 input file로 하여, 프로그램의 syntax tree를 출력하는 프로그램을 작성하시오.

Cute15의 built-in function

이번 과제에서 구현해야하는 built-in function들은 다음과 같다.

- 리스트 연산

car | cdr | cons

- 노드 연산

null? | atom? | eq?

1. car

List의 맨 처음 원소를 리턴한다.

```
> (car '(2 3 4))
```

```
2
```

```
> (car '((2 3) (4 5) 6))
```

```
(2 3)
```

주의: car는 list가 아닌 데이터나 ()이 인자로 주어지는 경우 error 를 내게 된다. 그러나, 본 과제에서는 이러한 error 발생 경우에 대해서는 고려하지 않고, 모든 입력이 올바르게 되어 주어진다고 가정한다. (이것은 car 외의 다른 함수에도 동일하게 적용한다.)

2. cdr

list의 맨 처음 원소를 제외한 나머지 list를 리턴한다. list가 아닌 데이터에 대해서는 error 를 낸다.

```
> (cdr '(2 3 4))
```

```
(3 4)
```

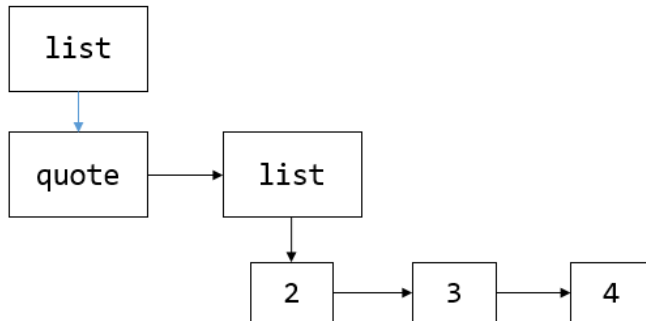
```
> (cdr '((2 3) (4 5) 6))
```

```
((4 5) 6)
```

```
> (cdr '(2))
```

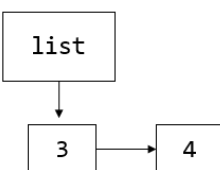
()

위의 두 `car`, `cdr`의 예를 그림으로 나타내면 다음과 같다.



이러한 노드가 있다고 하면

`car`는 

`cdr`는 

를 리턴한다.

3. `cons`

한 개의 원소(`head`)와 한 개의 리스트(`tail`)를 붙여서 새로운 리스트를 만들어 리턴한다.

```
> (cons 1 '(2 3 4))
```

```
(1 2 3 4)
```

```
> (cons (2 3) '(4 5 6))
```

```
((2 3) 4 5 6)
```

```
> (cons 2 ())
```

```
(2)
```

4. `null?`

리스트가 `NULL` 인지 검사한다. 즉, `()` 인지 검사한다.

```
> (null? ())
```

```
#T
```

```
> (null? '(1 2))
```

```
#F
```

```
> (null? '(()))
```

```
#F
```

5. atom?

list가 아니면 모두 atom 이다. 따라서 list인 경우는 false, list 가 아닌 경우는 true를 리턴한다. 주의 : **null list**은 **atom** 으로 취급된다.

```
> (atom? 'a)
```

```
#F
```

```
> (atom? '(1 2))
```

```
#F
```

```
> (atom? ())
```

```
#T
```

6. eq?

두 노드의 값이 같은지 비교한 값을 반환한다.

```
> (eq? 'a 'a)
```

```
#T
```

```
> (eq? 'a 'b)
```

```
#F
```

```
> (eq? '(a b) '(a b))
```

```
#T
```

수행 예시

입력파일

```
(car '(2 3 4))
```

```
(car '((2 3) (4 5) 6))
```

```
(cdr '(2 3 4))
```

```
(cdr '((2 3) (4 5) 6))
```

```
(cdr '(2))
```

출력파일

```
2
```

```
(2 3)
```

```
(3 4)
```

```
((4 5) 6)
```

```
()
```

Programming

car, cdr, cons, eq?, atom?, null?등을 처리하는 built-in 함수 처리를 위한 함수를 작성한다.

1. 노드의 자료구조

- 1) 각 노드마다 equals 메소드를 재정의 함: EQ? 의미에 맞게 정의
- 2) Node의 equals를 정의함으로써 next를 비교할 것

```
@Override
public boolean equals(Object obj){ ... }
```

IntNode 예시

```
@Override
public boolean equals(Object obj) {
    if(this==obj) return true;
    if(!(obj instanceof IntNode)) return false;
    if(!super.equals(obj)) return false;
    IntNode tmp = (IntNode) obj;
    if(this.value==tmp.value) return true;
    return false;
}
```

- 3) copyValue 메소드(Node의 추상함수) 추가 : input으로 받은 노드에 자신의 value를 추가

바뀐 Node

```
public abstract class Node {
    Node next;
    public Node() { this.next = null;}
    public void setNext(Node next){ this.next = next; }
    public void setLastNext(Node next){
        if(this.next != null) this.next.setLastNext(next);
        else this.next = next;
    }
    public Node getNext(){ return next; }

    @Override
    public boolean equals(Object arg0) {
        // TODO Auto-generated method stub
        //채워서 사용
    }
    public abstract void copyValue(Node node);
}
```

IntNode 예시

```
public void copyValue(Node node) {
    // TODO Auto-generated method stub
    this.value = ((IntNode) node).value;
}
```

2. 여러 명령어를 처리하도록 프로그램 작성
3. Built-in 함수 구현

```
public class CuteInterpreter {

    private final static BooleanNode TRUE_NODE = new BooleanNode();
    private final static BooleanNode FALSE_NODE = new BooleanNode();
    static {
        TRUE_NODE.value = true;
        FALSE_NODE.value = false;
    }

    private void errorLog(String err) {
        System.out.println(err);
    }

    enum CopyMode {
        NO_NEXT, NEXT
    }

    private Node runFunction(FunctionNode func) {
        Node rhs1 = func.getNext();
        Node rhs2 = (rhs1 != null) ? rhs1.getNext() : null;

        switch (func.value) {
            case CAR: {
                if (!checkQuote(rhs1))
                    errorLog("Syntax Error!");
                Node item = runQuote((ListNode) rhs1);

                //copyNode를 이용하여 값을 return 하시오.
            }
            case CDR: {
                if (!checkQuote(rhs1))
                    errorLog("Syntax Error!");
                Node item = runQuote((ListNode) rhs1);
                //copyNode를 이용하여 값을 return 하시오.
            }
            case CONS: {
                Node head = runExpr(rhs1);
                Node tail = runExpr(rhs2);

                //CONS에 맞게 작성할것

                return result;
            }
            case ATOM_Q:
                if (!(rhs1 instanceof ListNode)
                    || (((ListNode) rhs1).value == null))
                    return TRUE_NODE;
                else
                    return FALSE_NODE;
            case EQ_Q:
                if (rhs1 != null && rhs1.equals(rhs2)) {
                    return TRUE_NODE;
                } else {
                    return FALSE_NODE;
                }
            case NULL_Q:
                if (rhs1 instanceof ListNode && ((ListNode) rhs1).value == null)
                    return TRUE_NODE;
        }
    }
}
```

```

        else
            return FALSE_NODE;

    default:
        break;
    }

    return null;
}

private Node copyNode(Node node, CopyMode mode) {
    //node를 복사
    //mode에 따라서 next를 복사함
    if (node == null) return null;

    Node result = null;

    if (node instanceof BinarayOpNode) result = new BinarayOpNode();
    else if (node instanceof BooleanNode) result = new BooleanNode();
    else if (node instanceof FunctionNode) result = new FunctionNode();
    else if (node instanceof IdNode) result = new IdNode();
    else if (node instanceof IntNode) result = new IntNode();
    else if (node instanceof ListNode) result = new ListNode();
    result.copyValue(node);
    if (mode == CopyMode.NEXT && result != null)
        result.setNext(copyNode(node.getNext(), mode));

    return result;
}

private Node runList(ListNode list) {
    //list의 value가 QuoteNode일 경우
    if (list.value instanceof QuoteNode)
        return runQuote(list);
    Node opCode = list.value;

    if (opCode == null) return list;
    if (opCode instanceof FunctionNode)
        return runFunction((FunctionNode) opCode);

    return list;
}

private Node runQuote(ListNode node) {
    //QuoteNode의 value를 반환함
    QuoteNode qItem = (QuoteNode) node.value;
    Node item = qItem.value;

    return item;
}

public Node runExpr(Node rootExpr) {
    if (rootExpr == null)
        return null;

    if (rootExpr instanceof IdNode)
        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);
    else
        errorLog("run Expr error");
}

```

```

        return null;
    }

    private boolean checkQuote(Node node) {
        // QuoteNode의 형태인지 확인하는 메소드
        if (!(node instanceof ListNode))
            return false;
        ListNode tmp = (ListNode) node;
        if (!(tmp.value instanceof QuoteNode))
            return false;
        return true;
    }
}

```

4. 테스트

```
public static void main(String[] args) throws Exception {
    read file..
    parsing..
    print...
    CuteInterpreter i = new CuteInterpreter();
    Node resultNode = i.runExpr(node);
    print..
    ...
}
```