PL Assignment #8: Cute14 Project

과제물 부과일 : 2014-05-20(화)
제출 기한 : 각 아이템마다 다름

문제

그 동안 과제를 수행해 온 Cute14 문법에 따른 프로그램을 입력하면 결과를 출력하는 인터프리터를 설계 및 구현한다.

팀구성

- 개인이나 조 단위로 수행(2명)
  - **Item1은 개인수행**과제이며, **Item2-3은 개인이나 조 단위** 수행 과제임.
- 개인이 아닌 **조별로** 과제를 수행할 경우 **SVN을 반드시 이용**하여야 하며, **가산 점이 부여**된다.
- 실제 활동하지 않는 조원이 발생하면 해당 조원을 배제하고 개별 과제로 변환할 수 있다. 이와 같이 조를 다시 구성할 필요가 있는 경우에는 반드시 조교에게 알린다.

Project 내용

아래와 같이 3개의 item으로 나뉜다. 아래 주어진 일정에 따라 수행하며, 보고서를 제출한다.
조를 구성한 경우 각 조원간의 역할 분담도 명확하게 되어야 한다.

Item 1. 프로그램의 interpretation 환경 구현
Item 2. 변수의 바인딩 처리 : define 문으로 변수를 정의하고 사용할 수 있도록 지원한다.
Item 3. 함수의 바인딩 처리 : define 문으로 함수를 정의하고 사용할 수 있도록 지원한다.

일정

- 2014년 5월26일(월) 23:59:59까지 **Item 1**에 대한 **보고서**를 제출한다.
- 2014년 6월 9일(월) 23:59:59까지 **Item 2**에 대한 **보고서**를 제출한다.
- 2014년 6월18일(수) 23:59:59까지 **Item 3**에 대한 **보고서**를 제출한다. (option)

- 각 단계마다 프로그램 소스파일도 함께 업로드 해야 한다.
- 조 단위 수행인 경우 제출시 조원 중 한 사람이 업로드 한다. 또한 과제 **제출방법 공지사항을** 다시 한번 확인하여 보고서 내용 중 **조별관련 사항을 빠짐없이 작성**한다.
- 각 단계마다 별도의 **제출 지연은 불허**하며, 미완성하였더라도 제출시 일부 점수 반영.
- 배점은 item1:item2:item3가 1:3:2 가량이 되도록 할 예정임.

Item 1. 프로그램의 interpretation 환경(개인별 과제)

- Interpreter를 구동시키면 >나 $ 등의 적당한 prompt 를 띄운다.
- Prompt 다음에 사용자(프로그래머)가 Cute Expression을 입력하면 해당 값을 다음 줄에 출력한다. 출력 전에는 … 등의 표시 후 출력하여 가독성을 높인다.
- 예)

```
> (+ 1 3)
… 4
>
```

Item 2. 변수의 바인딩처리(개인별 또는 조별 수행)

- **변수에 대한 define 문을 처리한다.**
  - 예를 들어 (define a 1) 이라고 하면, a 의 값이 1 임을 저장하는 심벌테이블을 만든다. 간단한 방법으로 id와 노드를 원소로 하는 클래스를 만들어, 이에 대한 배열을 구성하는 것이 있다.
  - (define b '(1 2 3)) 등도 마찬가지로 처리한다. 즉, 테이블에 b가 '(1 2 3) 임을 저장한다.void insertTable(String id, Node value) 와 같은 함수를 만들어 사용한다.
  - (define c (- 5 2)) 와 같은 경우도 가능하다. 이때에는 insertTable에 삽입하기 전에 runExpr을 통해서 (- 5 2) 대신 결과값 3으로 바꾸어 저장한다.
  - 결과적으로 quoted 된 리스트나 상수만 테이블에 저장된다고 가정한다.
  - lambda 구문은 처리하지 않는다. (Item 2 까지는 없다고 가정한다.)

- **변수의 값을 사용할 수 있도록 한다.**
  - 예를 들어 (+ a 3) 은 먼저 a의 값인 1을 꺼내온다.
    Node lookupTable(String id) 과 같은 함수를 만들어 사용한다.a 대신 1을 사용하므로 결과로 4를 얻는다. (+ a 3)에서 a 대신 loopkupTable()의 결과로 대체한 후 runExpr()의 인자로 넣고 수행하여 결과로 4를 얻는다.
  - 같은 방식으로 (car b) 는 1 이 된다.
  - 오류가 있는 프로그램은 없다고 가정한다. (즉, 예를 들어 (+ b 1) 등의 input은 없다고 가정한다.)
  - 변수는 전역으로 정의된다고 가정한다.
  - 같은 변수를 두 번 이상 define 하면 앞에 define 된 것이 없어진다고 가정한다.

Item 3. 함수의 바인딩처리 (Option) (개인별 또는 조별 과제)

- **함수의 정의도 define 문으로 가능하므로 변수 바인딩과 함께 테이블에 저장한다.**
  - 예를 들어 plus1이 아래와 같이 정의되어 있다면, plus1이 (lambda …) 임을 테이블에 저장한다.

    ```
    (define plus1 (lambda (x) (+ x 1)))
    ```

  - 이와 같은 방식으로 구현할 수도 있지만, 함수의 인자 x를 따로 뽑아내어 테이블에 분리 저장함으로써 추후 호출 시간을 단축시킬 수 있다. 이러한 경우 별도의 추가점은 없다.
  - 변수와 마찬가지로 함수도 전역으로 정의된다. (즉, 중첩 정의 되지 않는다.) **단, 수업 중 배운 심벌테이블 처리 방법 중 하나를 사용해서 처리한다면 추가점이 있다.**
  - 함수 내에 변수 선언(즉, 지역변수 선언)은 없다고 가정한다. **단, 수업중 배운 심벌테이블 처리 방법 중 하나를 사용해서 처리한다면 추가점이 있다.**

- **함수 호출을 처리한다. (이하는 처리 예임)**
  - '(다음에 나오는 첫번째 원소는 그 다음에 두번째 원소, 즉 인자가 있을 때만 현재와 같이 수행한다. (기존 과제 ~07 과 동일)
    1. (car '(a b c)) ➔ a
    2. (+ 1 2) ➔ 3
  - '(다음에 나오는 첫번째 원소 다음에 인자가 없다면, 즉 '(내부에 인자가 한 개라면, 리스트가 그대로 리턴된다. (**기존 과제 ~07에 추가해야 하는 사항**)
    1. (car) ➔ (car)
    2. (+) ➔ +
  - '(다음에 나오는 첫번째 원소가 lambda 일 때
    1. 두번째 이후의 원소가 없다면 그대로 리턴된다.
       - (lambda (x) (+ x 1)) ➔ (lambda (x) (+ x 1))
    2. 두번째 이후의 원소가 있다면 아래와 같이 수행한다.
       - lambda 키워드 다음에 오는 (x) 의 x에 actual parameter를 바인딩하여 임시로 변수 테이블에 넣어둔다.

         eg. ((lambda (x) (+ x 1)) 10) 이라면 x에 10을 바인딩
       - lambda (x) 다음에 오는 리스트가 함수의 내용이므로 이것을 expression으로 생각하고 수행한다.

         eg. (+ x 1)
       - 함수 수행 후에는 임시로 x에 10을 바인딩 한 부분을 테이블에서 제거한다.
  - '(다음에 나오는 첫번째 원소가 일반 identifier 일때
    1. 인자가 없다면 리스트 자체가 그대로 리턴된다.
       - (a) ➔ a
    2. 인자가 있다면 identifier에 바인딩 된 함수, 즉 lambda 식을 가져와서 해당 인

자에 수행한다. 예를 들어 (plus1 10) 과 같은 expression이 입력되면, plus1에 해당되는 함수 정의인 lambda 식을 받아온 다음 아래와 같이 수행한다.

eg.

```
> (define plus1 (lambda (x) (+ x 1)) )
>(plus1 10)              ;; ((lambda (x) (+ x 1)) 10) 과 동일
… 11
```

■ 함수내에서 다른 함수 호출은 없다고 가정한다.

- 함수 내에서 다른 함수를 호출하도록 허용하려면, 인자를 처리하는 부분을 별도의 stack으로 구현하거나, interpreter의 runLambda 함수의 recursion을 적절히 활용하여 구현할 수 있다. 이러한 경우에는 추가점이 있다.

* Item3 는 option이므로 제출하지 않아도 된다. 제출 하더라도 구현 사양에 따라 추가점이 있으므로 document를 명확히 해야 불이익이 없다.