# PL Assignment Project: Cute19 Project

과제물 부과일 : 2019-05-22 (수)
Program Upload 마감일 : 각 아이템에 따라 다름

## 문제

그 동안 과제를 수행해 온 Cute 19문법에 따른 프로그램을 입력하면 결과를 출력하는 인터프리터를 설계 및 구현하라

## 팀 구성

- 개인이나 조 단위로 수행(최대 3명)
- 과제 진행 중 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 조원 간의 역할 분담도 보고서에 명확하게 명시 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.
Lambda 함수 구현 및 추가 기존 프로그램 발전

### 일정

- 2019년 6월02일(일) 23:59:59까지 **Item 1**에 대한 **보고서**를 제출한다.
- 2019년 6월04일(화) 1차 팀 공지(향후 팀을 만들 수는 없으며 개인팀전환만 가능)
- 2019년 6월09일(일) 23:59:59까지 **Item 2**에 대한 **보고서**를 제출한다.
- 2019년 6월19일(수) 23:59:59까지 **Item 3**에 대한 **보고서**를 제출한다. (option)

**Item 1.** 프로그램의 Interpretation환경(REPL : Read-Eval-Print Loop) 구현 **(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 가 독성을 높인다.(option)
- 기존에 사용하던 파일 구조를 수정하는 것.
- 보고서에는 결과 스크린 샷과 수정한 코드 부분을 첨부한다.
- 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.
  - `(define b '(1 2 3))` 등도 마찬가지로 처리한다. 즉, 테이블에 b가 `'(1 2 3)` 임을 저장한다. `insertTable(id, value)` 와 같은 함수를 만들어 사용한다.
  - `(define c (- 5 2))` 와 같은 경우도 가능하다. 이때에는 `insertTable`에 삽입하기 전에 `runExpr`을 통해서 `(- 5 2)` 대신 결과값 `3`으로 바꾸어 저장한다.
  - 결과적으로 quoted 된 리스트나 상수만 테이블에 저장된다고 가정한다.
  - 6월3일 추가 : Boolean 노드도 define 할 수 있다. – 각 과제 공지 참조

- **변수의 값을 사용할 수 있도록 한다.**
  - 예를 들어 `(+ a 3)` 은 먼저 `a`의 값인 `1`을 꺼내 온다.
    `Node lookupTable(String id)` 과 같은 함수를 만들어 사용한다.`a` 대신 `1`을 사용하므로 결과로 `4`를 얻는다. `(+ a 3)`에서 `a` 대신 `loopkupTable()`의 결과로 대체한 후 `runExpr()`의 인자로 넣고 수행하여 결과로 `4`를 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 및 코드 발전(Option) (개인별 또는 조별 과제)

- **함수의 정의도 define 문으로 가능하므로 변수 바인딩과 함께 테이블에 저장한다.**
  - 예를 들어 plus1이 아래와 같이 정의되어 있다면, `plus1`이 `(lambda …)` 임을 테이블에 저장한다.

    ex)

    ```
    (define plus1 (lambda (x) (+ x 1)))
    (plus1 2)
    … 3
    ```

  - 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.)
  - 함수 내에 변수 선언(즉, 지역변수 선언)은 없다고 가정한다.

- **Lambda 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 원소가 `lambda` 일 때

    1. 두번째 이후의 원소가 없다면 그대로 `return` 된다.
       - `(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.

         e.g. `((lambda (x) (+ x 1)) 10)` 이라면 `x`에 `10`을 바인딩
       - `lambda (x)` 다음에 오는 리스트가 함수의 내용이므로 이것을 `expression`으로 생각하고 수행한다.

         e.g. `(+ x 1)`
       - 함수 수행 후에는 임시로 x에 10을 바인딩 한 부분을 테이블에서 제거한다.

- **위의 항목을 제외하고 각각의 아이디어를 적용해 기존에 작성하던 과제 코드(인터프리터)를 발전시킨다. 각각의 경우 문서에 상세히 기록한다. (안되는 부분과 발전시킨 부분 등을 상세히 기재, 아래는 코드 발전 예시이다.)** (추가 구현시, 아주 조금 가산점 부여)
    - Lambda 구현
    - 함수 Define 구현
    - 전역 함수 호출
    - 함수 내에서 전역 함수 호출
    - 변수 scope 구현 (추가 구현)
    - Nested 함수 구현 가능 (추가 구현)
    - Recursion 함수 구현 가능 (추가 구현)
    - 함수에서 함수를 인자로 사용가능 (추가 구현)
    - 기타 기존에 하지 못했던 예외 처리 등. (추가 구현)

코드 발전 예시

```
((lambda (x) (+ x 1) ) 2)              #lambda 구현
> 3
(define plus1 (lambda (x) (+ x 1)))   #전역 함수 구현
(plus1 3)                              #전역 함수 구현
> 4
(define plus2
  (lambda (x) (+ (plus1 x) 1)))          #함수 내에서 전역 함수 호출 가능
(plus2 3)
> 5
(define cube
  (lambda (n)
   (define sqrt
      (lambda (n) (* n n)))
  (* (sqrt n) n)))                     #Nested 함수 구현

(define lastitem
  (lambda (ls)
     (cond
       ((null? (cdr ls)) (car ls))
       (#T (lastitem (cdr ls))))))    #Recursion 구현

  #예에서는 가 독성을 위해 엔터를 주었으나 구현 시 한 줄로 입력 받게 구현.
```

# 과제 제출 방법 및 유의사항

## ➢ Project 전체 유의사항

- 기존의 코드 수정 가능(코드 수정 여부 조교에게 물어보지 말 것.)
- 각 단계마다 프로그램 소스 파일도 함께 업로드 해야 한다. (프로젝트 업로드)
- 프로그램 비 동작 시 점수 없음(프로젝트 추가 설정 및 구동 환경이 특별하면 보고서에 기재, Item1 제외)
- 조 단위 수행인 경우 제출시 조원 중 한 사람이 업로드 한다. 또한 과제 **제출 방법 공지사항을** 다시 한 번 확인하여 보고서 내용 중 **조별 관련 사항을 빠짐없이 작성**한다.
- 각 단계마다 추가 제출은 없으며, 미완성하였더라도 제출시 일부 점수 반영. (Item1 제외)

## ➢ Item1

- Item1에 대한 내용은 그 동안 구현한 내용에 대한 입력으로 제출할 것
  Ex) (+ 1 2), (car '(2 3 4)) 등
- **Item1을 진행하지 않을 경우 향후 Item에 대하여 진행 할 수 없음(조원 중 한명이라도 Item1을 진행하지 못했을 경우 해당 조는 향후 과제 제출 무효.)**
- Item1을 제출 할 때 향후 조로 진행할 팀원을 보고서에 추가하여 제출 할 것.

## ➢ Item2

- 보고서와 소스코드를 압축하여 기존의 이러닝 사이트를 통해 제출(조장만)
- 보고서에는 팀원 정보 기재할 것(학번, 이름, 팀장, 팀 번호)
- 구현 함수 및 구현 방법과 알고리즘 상세히 기재

## ➢ Item3

- 보고서와 소스코드를 압축하여 기존의 이러닝 사이트를 통해 제출(조장만)
- 보고서에는 팀원 정보 기재할 것(학번, 이름, 팀장, 팀 번호)
- Lambda와 함수 define은 필수로 구현.
- 구현 함수 및 구현 방법과 알고리즘 상세히 기재
- 추가 발전 부분은 기존에 안되던 부분과 비교하여 보고서에 상세히 기재.