PL Assignment #08 : Cute17 Built-in Function 구현(2) 보고서

컴퓨터공학과
201504280 신윤호

1. 문제 해결 방법

• 산술 연산

과제에 참고자료로 주어진 내용을 참조해 run_binary 함수를 정의하였다. run_binary함수는 노드를 인자로 받아서 이항 연산을 수행하는 함수이다. left에는 좌항, right에는 우항이 저장되고, 연산자에 따라 left와 right에 적절한 연산을 수행한 후 그 결과값을 value로 갖는 노드를 반환하도록 구현했다.

minus, plus, multiple, divide 네 개의 산술 연산은 run_binary 함수를 호출하여 결과로 반환된 노드를 반환한다. 가령 (+ 1 2) 리스트의 경우 plus 함수에 전달되어, run_binary 함수로 전달된다. left의 value는 1, right의 value는 2 값을 가지게 되고, 연산자 노드의 타입이 PLUS이므로 덧셈 연산을 수행한다. 이때 노드가 가진 value는 문자이므로 int로 형변환을 수행한 후 연산한다. 연산이 끝나면 연산의 결과값을 value로 갖는 노드를 생성해 반환한다.

• 관계연산

관계 연산도 이항 연산이므로, 앞서 정의한 run_binary함수를 활용하였다. 산술 연산과 마찬가지로 left에는 좌항, right에는 우항이 저장된다. 그리고 연산자 노드의 타입에 따라 LT, GT, EQ중 적절한 관계 연산을 수행한다. 그리고 그 결과에 따라 True 또는 False노드를 반환한다. 가령, (< 1 5)인 경우 left의 value는 1이고, right의 value는 5이다. 1 < 5는 참이므로, True 노드를 반환한다.

• 논리연산

not_op함수는 먼저 run_expr(node.value.next)를 통해 피연산자가 True인지, False인지를 파악해 condition에 저장하도록 했다. 그리고 이 condition이 만약 True라면 반대로 False노드를 반환하고 False라면 True노드를 반환하도록 구현하였다. 만약 condition이 논리값이 아니라면 오류 문구를 출력하도록 했다.

• 조건문

cond함수는 run_cond함수를 호출하는 역할을 수행하고, 실질적인 조건문의 처리는 run_cond함수에서 진행된다. run_cond함수는 condition과 result를 가진다. condition은 순환 호출 과정 중 현재 단계에서 검사할 조건을 담는 객체이다. 그리고 result는 condition이 true일 경우 반환할 노드를 담는 객체이다. 더 구체적으로 이야기 하자면, condition에는 node.value가 대입되고, result에는 condition.next가 대입된다. 예를 들어, (cond ((> 100 10) 2) 일 경우, condition 은 (> 100 10)이고, result의 value는 2이다. 100 > 10 이므로 condition은 True이고 따라서 이 경우 result가 반환된다.

만약 condition이 False일 경우에는 node.next에 대해 run_cond함수를 순환 호출한다. 이를 통해 두 번째 조건을 condition으로 하여 조건을 검토할 수 있다. 가령 (cond (#F 1) (#T 2))일 경우, 첫 번째 호출에서 condition은 False이다. 따라서 run_cond함수를 순환 호출하게 된다. 두 번째 호출에서 condition은 True이고, 따라서 value가 2인 노드를 반환하게 된다.

혹시 모든 조건이 False라면, 즉 node.next == None인 경우까지 True가 나오지 않았다면 순환호출을 멈추고 None을 반환한다.

2. 느낀점

저번 과제와 이어지는 과제라서 저번 과제를 할 때 정리해놓은 것들을 활용할 수 있었다. 지난번 과제때 코드 읽느라 엄청 힘들었는데, 덕분에 이번 과제는 조금 편한 마음으로 할 수 있었다. 그리고 파이썬 코드는 아니었지만 run_binary의 코드가 주어진 것도 큰 도움이 되었다. 또 cond를 C랑 비슷한 문법으로 표현해준 것도 도움이 되었다. 조금 놀랐던건 주어진 runBinary코드에 switch문이 있어서 자연스럽게 파이썬에서 switch문을 쓰려고 했는데, 파이썬에는 switch문이 없다는 것을 알게 되었다. 찾아보니까 딕셔너리를 써서 switch문과 비슷하게 구현할 수 있다고 하는데, 잘 이해되지 않아서 그냥 if문을 사용했다.

3. 테스트 코드 실행 결과

```
def Test_All():
    Test_method("(+ 1 2 )")
    Test_method("(- ( + 1 2 ) 4 )")
    Test_method("(* 3 2 )")
    Test_method("(/ 10 2 )")
    Test_method("(< 1 5 )")
    Test_method("(= 3 ( + 1 2 ) )")
    Test_method("(> 1 5 )")
    Test_method("(not #F )")
    Test_method("(null? '( 1 2 3) )")
    Test_method("(cond (#F 1) ( #T 2 ) )")
    Test_method("(cond ( ( null? ' ( 1 2 3 ) ) 1 ) ( ( > 100 10 ) 2 ) ( #T 3 ) )")

Test_All()
```

```
C:₩Python27₩python.exe "C:/Users/Yoonho/Google 드라이브/대학/프로그래밍언어/과제/hw08/hw08/hw08 Frame.py"
3
-1
6
5
#T
#T
#F
#T
#F
2
2

Process finished with exit code 0
```