EECS3311 Software Design (Fall 2020)

Q&A - Lab1

Friday, September 18

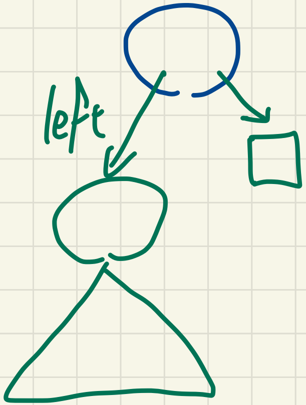
- Extra Scheduled Labs
- Lab0 (implementation)
- Lecture Series W2 (postcondition)
- Lab1
  * instructions
  * starter project
  * tutorial videos

visual

splay

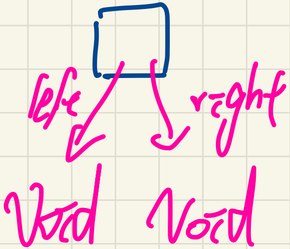
Internal

not void

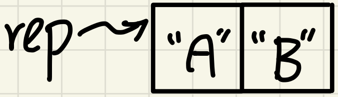


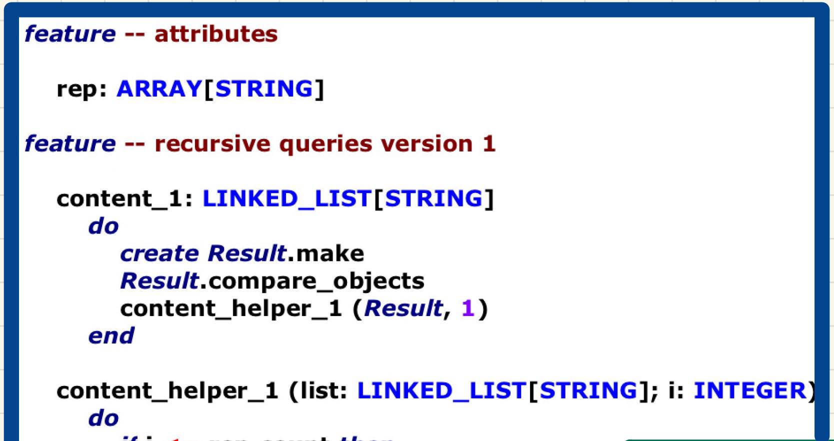
External

not void

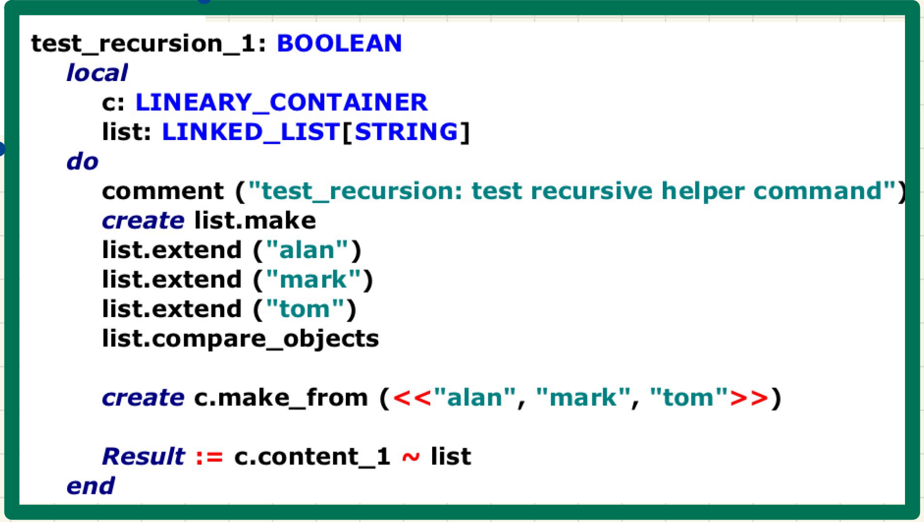


Recursive Helper Command





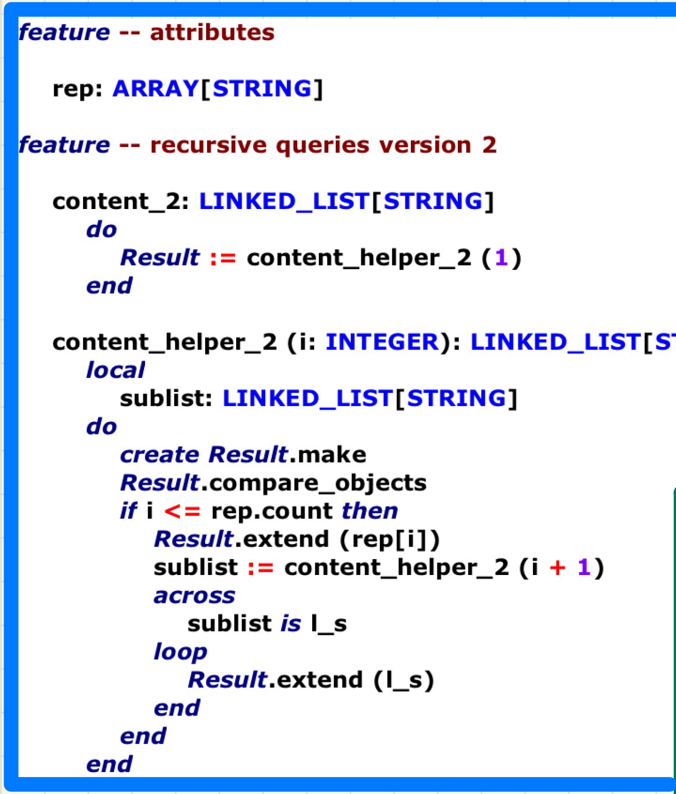
```
feature -- attributes

   rep: ARRAY[STRING]

feature -- recursive queries version 1

   content_1: LINKED_LIST[STRING]
      do
         create Result.make
         Result.compare_objects
         content_helper_1 (Result, 1)
      end

   content_helper_1 (list: LINKED_LIST[STRING]; i: INTEGER)
      do
         if i <= rep.count then
            list.extend (rep[i])
            content_helper_1 (list, i + 1)
         end
      end
```



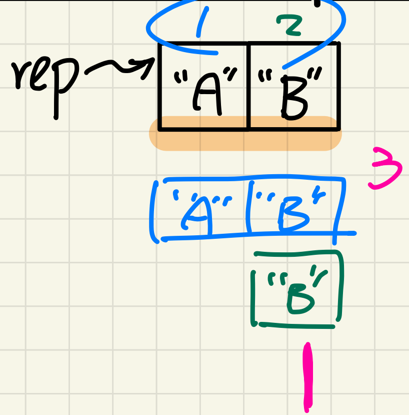
```
test_recursion_1: BOOLEAN
   local
      c: LINEARY_CONTAINER
      list: LINKED_LIST[STRING]
   do
      comment ("test_recursion: test recursive helper command")
      create list.make
      list.extend ("alan")
      list.extend ("mark")
      list.extend ("tom")
      list.compare_objects

      create c.make_from (<<"alan", "mark", "tom">>)

      Result := c.content_1 ~ list
   end
```

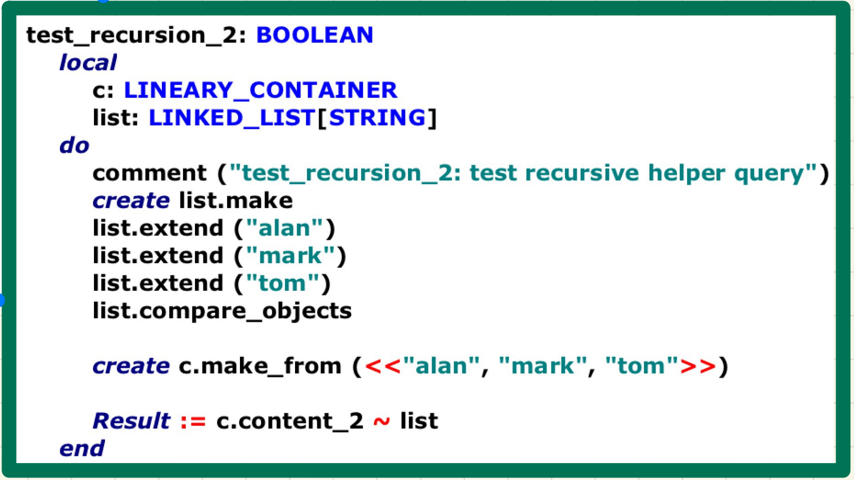
content_1
content_helper_1(list, 1)
content_helper_1(list, 2)
content_helper_1(list, 3)

Recursive Helper Query



```
feature -- attributes

   rep: ARRAY[STRING]

feature -- recursive queries version 2

   content_2: LINKED_LIST[STRING]
      do
         Result := content_helper_2 (1)
      end

   content_helper_2 (i: INTEGER): LINKED_LIST[STRING]
      local
         sublist: LINKED_LIST[STRING]
      do
         create Result.make
         Result.compare_objects
         if i <= rep.count then
            Result.extend (rep[i])
            sublist := content_helper_2 (i + 1)
            across
               sublist is l_s
            loop
               Result.extend (l_s)
            end
         end
      end
```



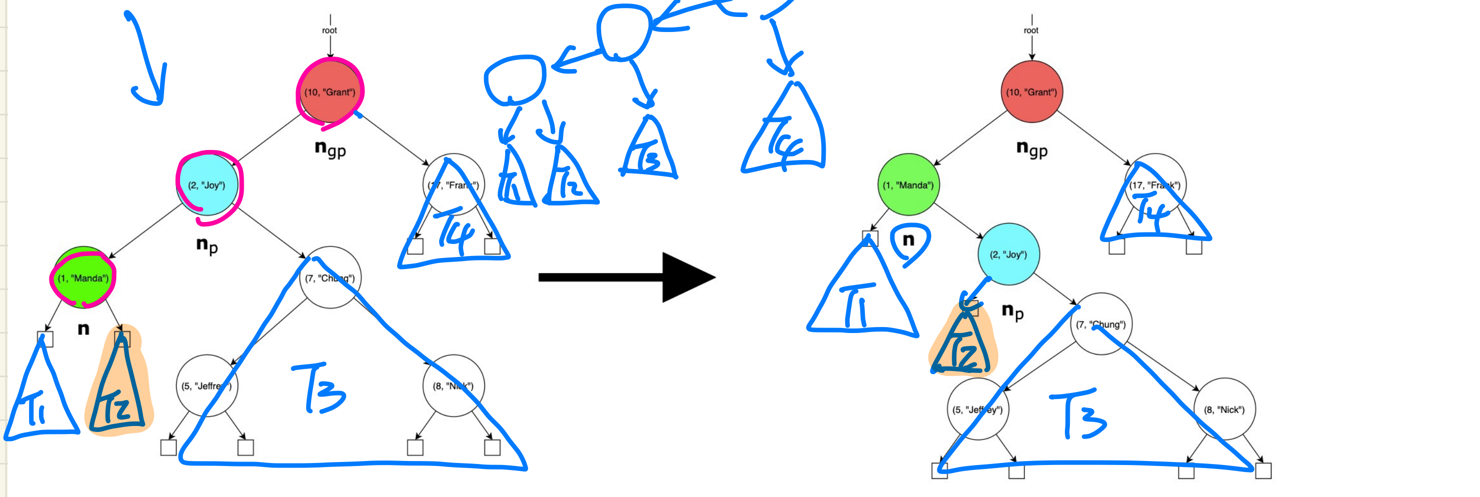


```
test_recursion_2: BOOLEAN
   local
      c: LINEARY_CONTAINER
      list: LINKED_LIST[STRING]
   do
      comment ("test_recursion_2: test recursive helper query")
      create list.make
      list.extend ("alan")
      list.extend ("mark")
      list.extend ("tom")
      list.compare_objects

      create c.make_from (<<"alan", "mark", "tom">>)

      Result := c.content_2 ~ list
   end
```

content_2
content_helper_2(1)
content_helper_2(2)
content_helper_2(3)

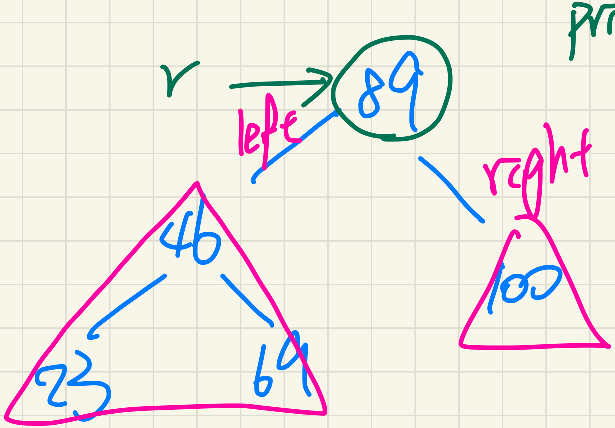
Understand Rotations

Case 2: n is the left child of n_p. Replace n_p with n. Then, set n's right child as n_p.



T_1 n T_2 n_p T_3 n_{gp} T_4

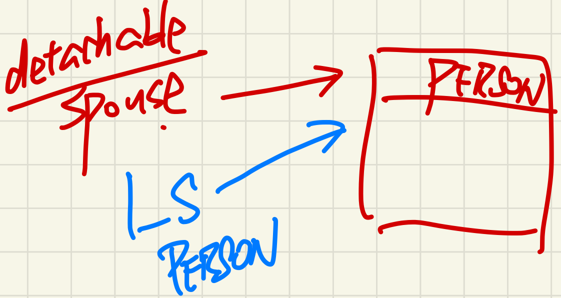
88



```
print (r)
  ↳ "89"            46   23   69
    print (r. left)
    print (r. right)       100
```

```
class PERSON

  spouse : detachable PERSON

invariant
  legal: Current.spouse.spouse = Current
end
```

might be void



① attached spouse as LS

↓ and then LS. spouse = Current

not appropriate

e.g. single person.

× spouse. spouse = Current

detachable

② attached spouse as LS

implies LS. spouse = Current

x : PERSON

y : detachable PERSON

cmd1 (p : PERSON) → p := y

↳ p := x ✓

cmd2 (p : detachable PERSON)

↳ p := y

p := x

det. att

can never be void

might be void

① x := y ✗

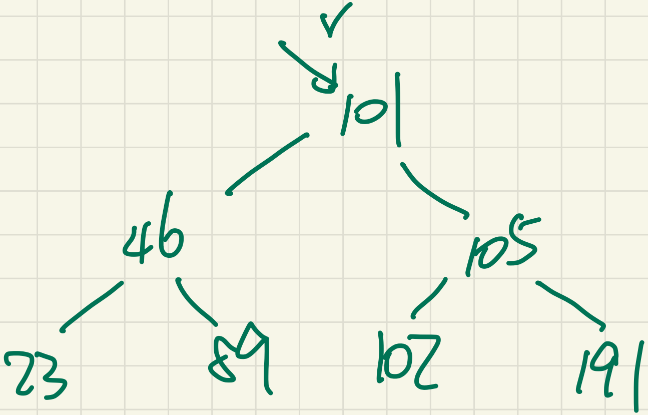
② y := x ✓

③ cmd1(x) ✓

④ cmd2(y)

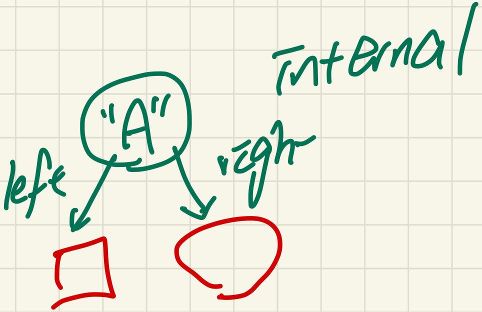
⑤ cmd1(y) X

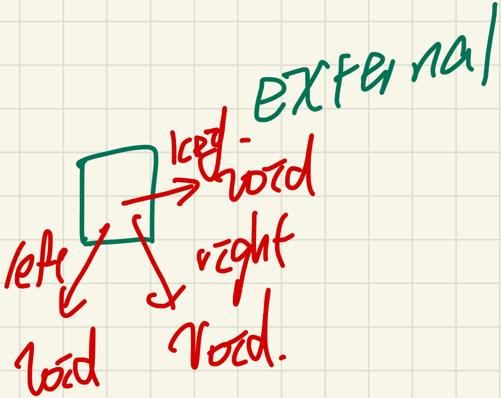
cmd2(x) →

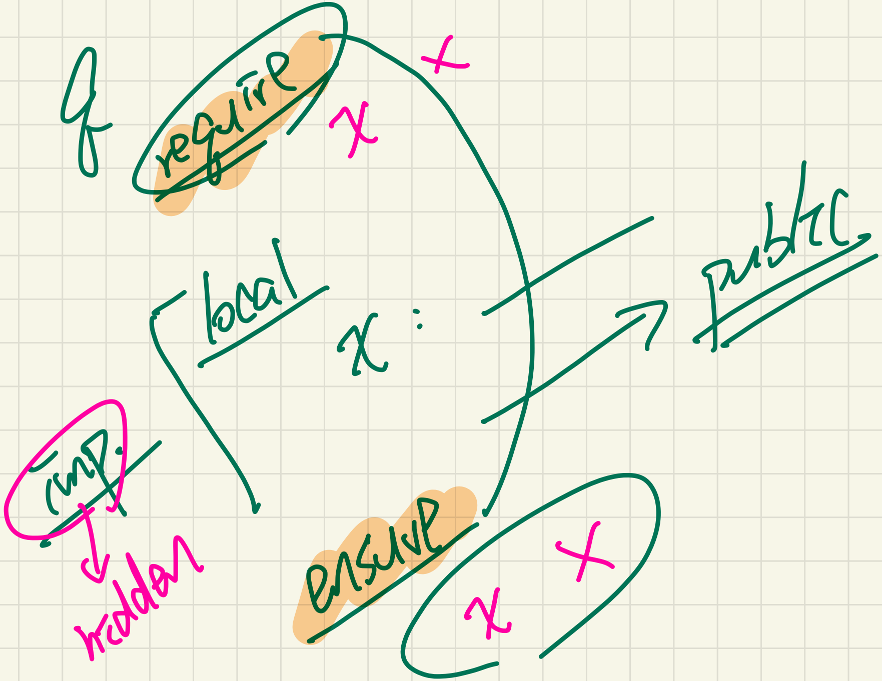


r.count 7

In-order : 23 46 89 101 102 105 191


internal
"A"
left
right


external
key -
void
left
right
void
void.


q
require
x
x
local
x :
public
imp.
hidden
ensure
x
x