Въпрос **1**

Отговорен

От максимално 1,00

Каква е областта на действие на локалните имена, дефинирани с помощта на специалната форма ***let**** в езика Racket?

Общият вид на обръщението към let* е следното:

(let* ([<пром-1><израз-1>]

.....

[<пром-n><израз-n>])

<тяло>)

След оценяване на дадена променлива и съответния ѝ израз, тя може да бъде използвана, както в изразите след нея, така и в тялото. Променливата не може да бъде използване в съответния си израз, т.е, пром-i , може да бъде използвана в изрази с номера j<=i+1<=n и тялото.

Въпрос **2**

Отговорен

От максимално 1,00

Обяснете понятието „рекурсивен изчислителен процес“. Дефинирайте функция на езика Racket, която генерира рекурсивен изчислителен процес, и обяснете предназначението на тази функция.

Рекурсивен изчислителен процес е такъв процес, който по време на на изпълнението си се обръща сам към себе си - сложната задача се манипулира до по-проста докато се достигне базов случай на задачата, като начинът за решение остава един и същ.

Пример за рекурсивен изчислитен процес е задачата за намиране на факториел на дадено число n.

Ще реализираме тази фунцкия с линейно итеративен рекурсивен процес.

```
(define (fact n)
   (define (helper current-n res)
      (if (> current-n n)      ->дъно на рекурсията
       res
      (helper (add1 current-n) (* current-n res))
      )
    (helper 1 1)  ->начални стойности на итеративния процес за факториел
   )
)
```

Тук res държи текущото състояние на резултата, а current-n следи докъде сме стигнали. Рекурсията се случва в процедурата helper.

Факториел може да се реализира и с линейно рекурсивен процес.

```
(define (fact-rec n)
  (if ( = n 1)  -> базов случай / дъно
        1
        (* n  (fact-rec (sub1 n)))
   )
)
```

Тук рекурсията се случва в самата процедура fact-rec - тя се обръща сама към себе си.

Имаме и процес на разгъване на опашката (всичките извиквания на fact-rec, в които ще се оценяват аргументе) и сгъване на опашката - всички аргументи са оценени и остава само да бъдат приложени процедурите.

Въпрос **3**

Отговорен

От максимално 1,00

Формулирайте общото правило за оценяване на комбинации в езика Racket.

1. Оценяват се всички подизрази.
2. Процедура, зададена чрез най-левият подизраз, се прилага върху оценките на всички подизрази, т.е. те са нейните аргументи (процедурите се оценяват "отвътре навън"). Оценката на комбинацията е оценката на най-левия подизраз.

Пример:

(+ (* 2 4) (/ 25 5)) -> (+ 8 5) -> 13

Първо се оценяват (* 2 4) и (/ 25 5) и техните оценки се явяват аргументи на процедурата +.

Въпрос **4**

Отговорен

От максимално 1,00

Дефинирайте понятието „списък“ в езика Racket.

1. Празният списък '() е списък.
2. Нека празният списък е a. Ако b е S-израз, то (b.a), както и (a.b), също е списък.

Въпрос **5**

Отговорен

От максимално 1,00

Обяснете действието на вградената функция **apply** в езика Racket. Дайте поне два примера.

Функцията apply прилага дадена процедура върху всеки един елемент на подаден списък.

Общ вид на обръщението към тази функция е:

(apply <процедура> <списък>)

Примери:

1. (apply (lambda (x) (add1 x) '(1 2 3 4 5))) ще върне като резултат '(2 3 4 5 6)
2. Нека имаме дефинирана следната процедура:

(define (sum-digits n)) -> която ще връща сумата на цифрите на дадено число.

Тогава (apply (lambda (x) (sum-digits x) '(12 34 25 108 546))) ще върне като резултат '(3 7 7 9 15)

Въпрос **6**

Отговорен

От максимално 1,00

Проследете процеса на оценяване и напишете оценката на всеки от следващите изрази на езика Racket:

(map (lambda (x) (list 1 x)) (list 1 2 3))

(map (lambda (x) (* x x)) ((lambda (y) (append '(1 2 3) y)) '(4 5)))

1. Нека разгледаме последователно процедурите.
(list 1 2 3) -> (1 2 3)

След това имаме lambda, приложена върху всеки елемент от този списък, чрез процедурата map. Тази lambda взима всеки елемент от списъка и на негово място поставя списък (1 елемент).

Следователно, крайният резултат ще е: ((1 1) (1 2) (1 3) .

2. Първо ще приложим lambda с аргумент y върху '(4 5) -> (1 2 3 4 5), защото процедурата append приема цял списък, а не елемент от него.

След това прилагаме lambda с аргумент x, който вече се отнася за всеки елемент от списъка.

Следователно, крайният резултат ще е: (1 4 9 16 25).

Въпрос **7**

Отговорен

От максимално 1,00

Как се осъществява от интерпретаторите на Racket прилагането на дадена съставна процедура към съответните аргументи съгласно модела на средите? Дайте пример.

Модела на средите не се различава по същество от нормалния модел за оценяване на комбинации.

Следва се общото правило:

- Първо се оценяват всички подизрази.
- След това се оценява най-левият подизраз, като неговите аргументи са оценките на следващите подизрази.Оценката на съставната процедура е оценката на най-левия подизраз.

Според модела на средите, за да се оцени една съставна процедура, трябва да бъде създадена таблица, в която се извършват свързванията между формалните аргументи и фактическите аргументи на процедурата, след което се оценява дефинията (тялото) на тази процедура, в което се използват вече оценените аргументи и оценката на самата процедура се получава в новополучената среда, която се е създала при извикването ѝ. В таблицата участва също и указател към родителската (съдържащата) среда на тази процедура. Ако в тялото на процедурата участва променлива, която не е била свързана в новопостроената среда, се търси свързване в обграждащите я. Ако такова не е намерено, тази променлива е несвързана(unbound) и тялото на фунцкията е некоректно.

Процедури съществено се оценяват само чрез lambda изрази. В тялото на процедурата се създава такъв lambda израз със същите аргументи като на началната процедура и неговата оценка е оценката на съставната процедура.

Пример:

```
(define (cube x)
    (* x x x)
)
```

е еквивалентно на

```
(lambda (x) (* x x x ) )
```

Въпрос **8**

Отговорен

От максимално 1,00

Нека е дадена (т.е. оценена от интерпретатора на Racket) следната дефиниция:

```
(define (f l)
  (cond [(null? l) '()]
        [(not (pair? (car l))) (cons (car l) (f (cdr l)))]
        [else (append (f (car l)) (f (cdr l)))]))
```

Проследете процеса на оценяване и напишете оценката на следващия израз:

```
(f '((2 (3 (4))) 5 ((6 (7)) 8)))
```

Нека разгледаме какво ще се случи:

Първо ще имаме f((2 (3 (4)))) f ((5 ((6 (7)) 8)). Да разгледаме лявото извикване на f.

(2 f((3 (4)))) -> (2 3 f((4))) -> (2 3 4 f('())) -> (2 3 4).

На стъпка три влизаме в условие номер 1 (списъкът е празен).

Аналогично и в дясното извикване на функцията.

Следователно, оценката на първоначалното извикване ще е : (2 3 4 5 6 7 8).

◄ Първо контролно върху задачи - предаване на решения (4та група)

Отиди на ...

Първо контролно по ФП - теоретична част ►