```
(define (zarb x y)
  (cond
   [(and (null? x) (null? y)) '()]
   [else (cons (* (car x) (car y))
               (zarb (cdr x) (cdr y)))]
  )
)

(define (reversed listt)
        (if (null? listt) '()
            (append (reversed (cdr listt))
                    (list (car listt)))))

(define (find_func x y)
        (zarb x (reversed y)))
```

تابع جواب ←

```
(define empty-q (lambda () '()))
```

```
(define add_to_end (lambda (lst symbl)
                     (append lst (list symbll))))
```

```
(define add_to_start (lambda (lst symbl)
                       (cons symbl lst)))
```

```
(define pop-1st-element (lambda (lst)
                          (cdr lst)))
```

```
(define pop-last-element (lambda (lst)
  (cond
    [(null? (cdr lst)) '()]
    [else (cons (car lst) (pop-last-element
                           (cdr lst)))]
  )
))
```

```
(define length (lambda (lst) (cond
  [(null? lst) 0]
  [else (+ 1 (length (cdr lst)))]
  )
))
```

ادامه‌ی سؤال ۲

صف دوطرفه و ویژگی هایش :

```
[0] = '()

(define 1st-elemnt (lambda (lst) (cond
                [(null? lst) error-string]
                [else (car lst)] )
         )
     )

(define last-elemnt (lambda (lst) (cond
        [(null? (cdr lst)) (car lst)]
        [else (last-elemnt (cdr lst))] )
    )
 )
```

③ تایپ استرینگ

ترمینال char را تعریف می‌کنیم (حروف الفبا و کاراکترهای اعداد)

$$char = \{"a", "b", \dots, "z"\ "A", "B", \dots, "Z"\ "1", "2", \dots, "9", "0"\}$$

① $$\frac{}{"" \in S}$$

② $$\frac{c \in char}{c \in S}$$

③ $$\frac{str1 \in S \quad str2 \in S}{str1 + str2 \in S}$$

+ : معنای پایتونی (append)

سؤال ④

```
Expression ::=  string
                const-str (str)

Expression ::=  append ( Expression , Expression )
                appnd-str (exp1, exp2)
```

→

```
(def appnd-str (lambda (x1) (x2)
    (string-append
        x1 x2 )
))
```

```
Expression ::= length ( Expression )
               len-str (exp1)
```

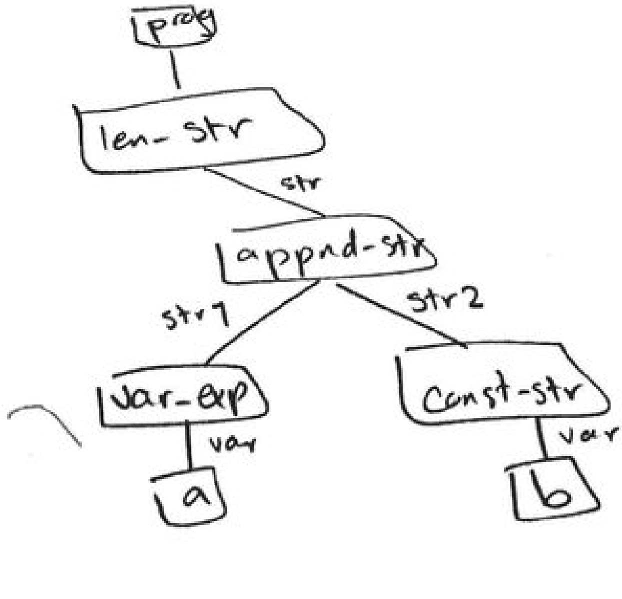
② سؤال length مثل ← len-str

define نمی‌شود

مثال: (scan&parse "length(append(a , "b"))"

```
# (struct : a-program
   #(struct : len-str
      #(struct : appnd-str
         #(struct : var-exp a )
         #(struct : const-str "b" )
      )
   )
)
```

concrete



abstract