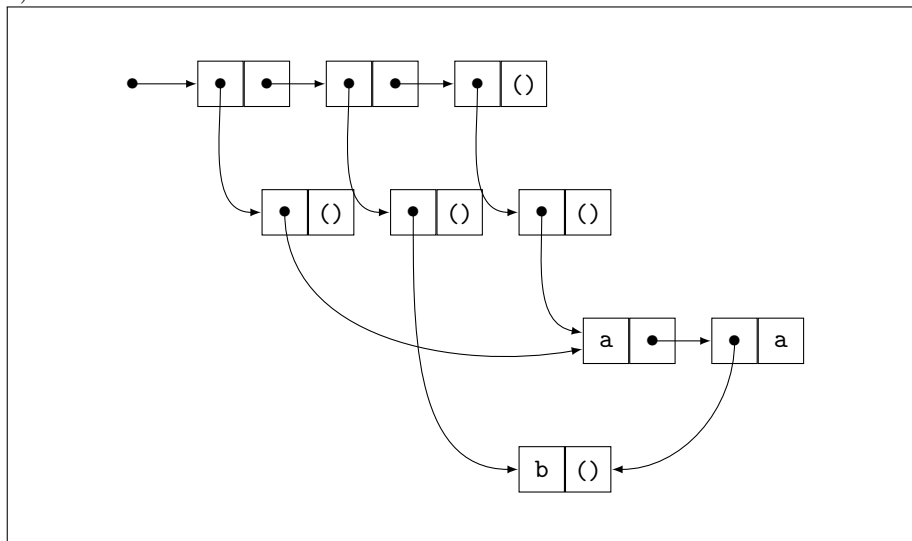
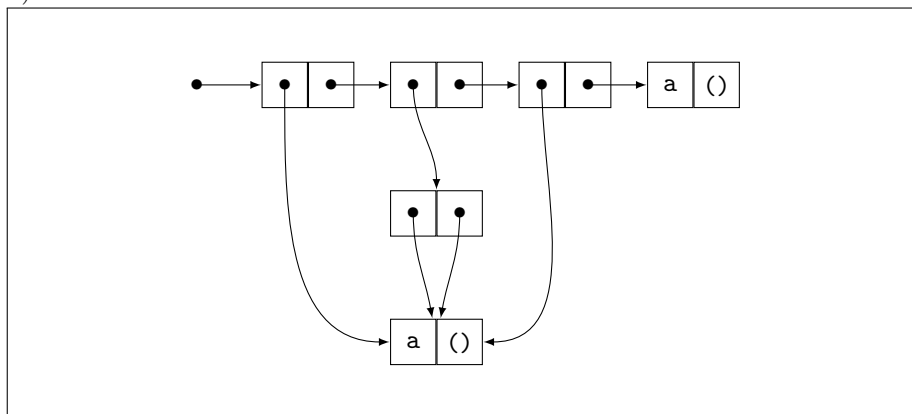


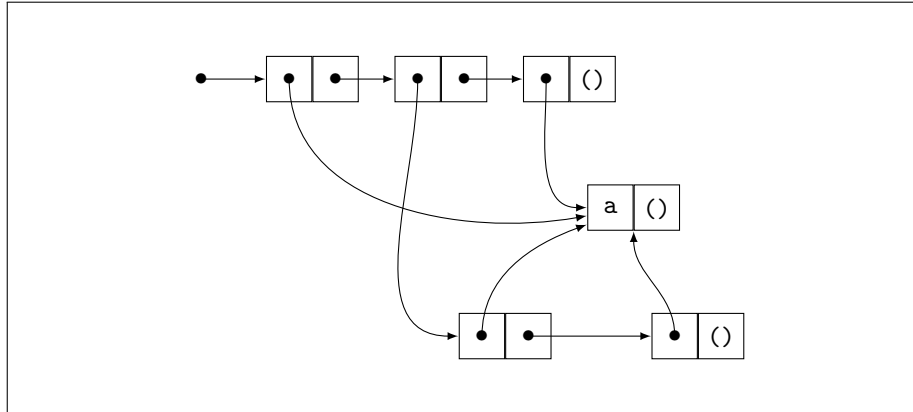
a)



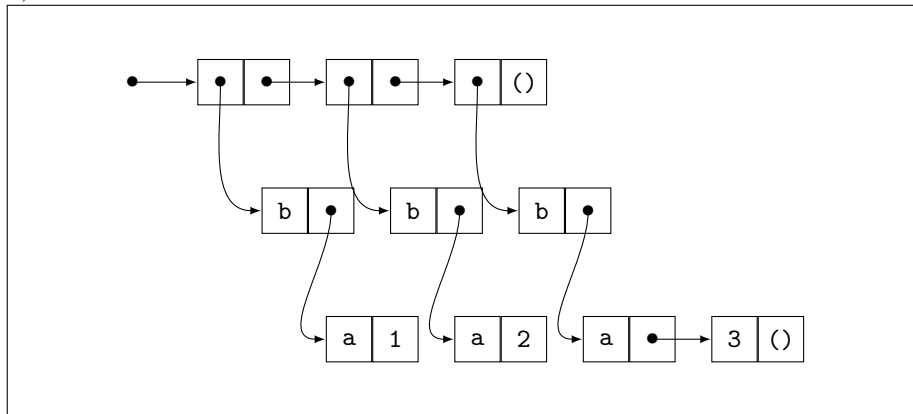
--	--



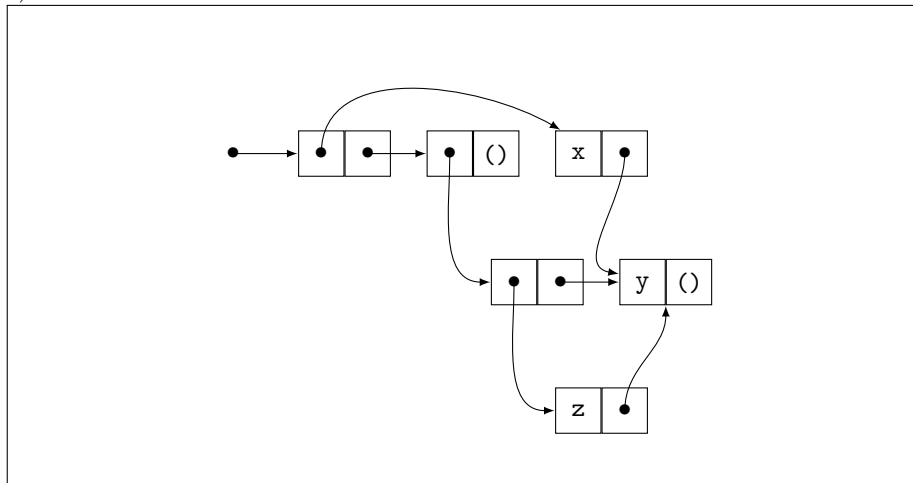
c)



d)



e)



```

graph LR
    Start(( )) --> Node1[ ]
    Node1 --> Node2[ ]
    Node2 --> Node3[ ]
    Node2 --> Node4[ ]
    Node3 --> Node5[ ]
    Node4 --> Node5
    Node5 --> End(( ))
    style Node1 fill:none,stroke:#000,stroke-width:1px
    style Node2 fill:none,stroke:#000,stroke-width:1px
    style Node3 fill:none,stroke:#000,stroke-width:1px
    style Node4 fill:none,stroke:#000,stroke-width:1px
    style Node5 fill:none,stroke:#000,stroke-width:1px

```

The diagram illustrates a linked list structure with three nodes. Each node is represented as a box divided into two parts: the left part for the data value and the right part for the next pointer. The first node contains the value '1' and its pointer points to the second node. The second node contains the value '2' and its pointer points to the third node. The third node contains the value '3' and its pointer points to a null value, represented by '()'. A separate pointer, indicated by a black dot, points to the first node.

The diagram shows a linked list with three nodes. Each node is a rectangle divided into two parts: the left part contains a black dot representing a pointer, and the right part contains data. The first node's pointer points to the second node. The second node's pointer points to the third node. The third node's pointer points to a fourth node (not shown). The second node's pointer also points to a fifth node (not shown). The first node's pointer also points to a sixth node (not shown).

Řešení:

- a) 

```
(let* ((x '(b))
      (y '(a ,x . a))
      (a1 (list y))
      (a2 (list x)))
      (list a1 a2 a1))
```
- b) 

```
(let* ((a '(a))
      (b (cons a a)))
      (list a b a 'a))
```
- c) 

```
(let* ((a '(a))
      (b (list a a)))
      (list a b a))
```
- d) 

```
(let* ((b1 '(b a . 1))
      (b2 '(b a . 2))
      (b3 '(b a 3)))
      (list b1 b2 b3))
```
- e) 

```
(let* ((y '(y))
      (z (cons 'z y))
      (x (cons 'x y))
      (a (cons z y)))
      (list x a))
```
- f) 

```
(let* ((s10 '(10))
      (ss10 (list s10))
      (ps105 (cons s10 5))
      (p (cons ps105 s10)))
      (list p ps105 ss10))
```
- g) 

```
(let* ((p23 (cons 2 3))
      (p (cons (list 3) p23)))
      (cons (cons 1 p23) p))
```
- h) 

```
(let* ((s10 '(10))
      (p (cons s10 s10)))
      (list s10 p p))
```

DODELAT: vysvetleni