Insert a node at a specific position in a linked list

locked

Problem Submissions Leaderboard Discussions

This challenge is part of a tutorial track by MyCodeSchool and is accompanied by a video lesson.

Given the pointer to the head node of a linked list and an integer to insert at a certain position, create a new node with the given integer as its $data$ attribute, insert this node at the desired position and return the head node.

A position of 0 indicates head, a position of 1 indicates one node away from the head and so on. The head pointer given may be null meaning that the initial list is empty.

Example
$head$ refers to the first node in the list $1 \rightarrow 2 \rightarrow 3$
$data = 4$
$position = 2$

Insert a node at position 2 with $data = 4$. The new list is $1 \rightarrow 2 \rightarrow 4 \rightarrow 3$

Function Description Complete the function *insertNodeAtPosition* in the editor below. It must return a reference to the head node of your finished list.

insertNodeAtPosition has the following parameters:

- *head*: a SinglyLinkedListNode pointer to the head of the list
- *data*: an integer value to insert as data in your new node
- *position*: an integer position to insert the new node, zero based indexing

Returns

- *SinglyLinkedListNode pointer:* a reference to the head of the revised list

Input Format

The first line contains an integer n, the number of elements in the linked list.
Each of the next n lines contains an integer SinglyLinkedListNode[i].data.
The next line contains an integer $data$, the data of the node that is to be inserted.
The last line contains an integer $position$.

Constraints

- $1 \leq n \leq 1000$
- $1 \leq SinglyLinkedListNode[i].data \leq 1000$, where $SinglyLinkedListNode[i]$ is the i^{th} element of the linked list.
- $0 \leq position \leq n$.

Sample Input

```
3
16
13
7
1
2
```

Sample Output

```
16 13 1 7
```

Explanation

The initial linked list is $16 \rightarrow 13 \rightarrow 7$. Insert 1 at the position 2 which currently has 7 in it. The updated linked list is $16 \rightarrow 13 \rightarrow 1 \rightarrow 7$.

Submissions: 8
Max Score: 50
Difficulty: Easy

Rate This Challenge:

More

Clojure

```
(definterface ISinglyLinkedListNode
    (getdata [])
    (getnext [])
    (setdata [node-data])
    (setnext [next-ptr]))

(deftype SinglyLinkedListNode [^:volatile-mutable data ^:volatile-mutable next]

    ISinglyLinkedListNode
    (getdata [_] data)
    (getnext [_] next)
    (setdata [_ node-data] (set! data node-data))
    (setnext [_ next-ptr] (set! next next-ptr)))

(definterface ISinglyLinkedList
    (gethead [])
    (gettail [])
    (sethead [list-head])
    (settail [list-tail])
    (insertnode [node-data]))

(deftype SinglyLinkedList [^:volatile-mutable head ^:volatile-mutable tail]
    ISinglyLinkedList

    (gethead [_] head)
    (gettail [_] tail)
    (sethead [_ list-head] (set! head list-head))
    (settail [_ list-tail] (set! tail list-tail))

    (insertnode [this node-data] [
        (def node (SinglyLinkedListNode. node-data nil))
        (if head [(.setnext (.gettail this) node)] (.sethead this node))
        (.settail this node)
    ])
)

(defn print-singly-linked-list [node sep fptr]
    (when node
        [
            (spit fptr (.getdata node) :append true)

            (when (.getnext node) (spit fptr sep :append true))

            (print-singly-linked-list (.getnext node) sep fptr)
        ]
    )
)

;
; Complete the 'insertNodeAtPosition' function below.
;
; The function is expected to return an INTEGER_SINGLY_LINKED_LIST.
; The function accepts following parameters:
;  1. INTEGER_SINGLY_LINKED_LIST llist
;  2. INTEGER data
;  3. INTEGER position
;


;
; For your reference:
;
; SinglyLinkedListNode [
;     ^:volatile-mutable data
;     ^:volatile-mutable next
; ]
;
;

(defn insertNodeAtPosition [llist data position]

)

(def fptr (get (System/getenv) "OUTPUT_PATH"))

(def llist-count (Integer/parseInt (clojure.string/trim (read-line))))

(def llist (SinglyLinkedList. nil nil))

(doseq [_ (range llist-count)]
    (.insertnode llist (Integer/parseInt (read-line)))
)

(def data (Integer/parseInt (clojure.string/trim (read-line))))

(def position (Integer/parseInt (clojure.string/trim (read-line))))

(def llist_head (insertNodeAtPosition (.gethead llist) data position))

(print-singly-linked-list llist_head " " fptr)
(spit fptr "\n" :append true)

```

Line: 25 Col: 1

Upload Code as File Test against custom input Run Code Submit Code