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.

You're given the pointer to the head node of a linked list, an integer to add to the list and the position at which the integer must be inserted. Create a new node with the given integer, 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.

As an example, if your list starts as $1 \to 2 \to 3$ and you want to insert a node at position 2 with $data = 4$, your new list should be $1 \to 2 \to 4 \to 3$

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

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 $data$ denoting 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.
- $0 \leq position \leq n$.

Output Format

Return a reference to the list head. Locked code prints the list for you.

Sample Input

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

Sample Output

```
16 13 1 7
```

Explanation

The initial linked list is `16 13 7`. We have to insert 1 at the position 2 which currently has 7 in it. The updated linked list will be `16 13 1 7`

Submissions: 134
Max Score: 25
Difficulty: Easy

Rate This Challenge:

More

Current Buffer (saved locally, editable) C++

```
#include ↔

using namespace std;

class SinglyLinkedListNode {
    public:
        int data;
        SinglyLinkedListNode *next;

        SinglyLinkedListNode(int node_data) {
            this->data = node_data;
            this->next = nullptr;
        }
};

class SinglyLinkedList {
    public:
        SinglyLinkedListNode *head;
        SinglyLinkedListNode *tail;

        SinglyLinkedList() {
            this->head = nullptr;
            this->tail = nullptr;
        }

        void insert_node(int node_data) {
            SinglyLinkedListNode* node = new SinglyLinkedListNode(node_data);

            if (!this->head) {
                this->head = node;
            } else {
                this->tail->next = node;
            }

            this->tail = node;
        }
};

void print_singly_linked_list(SinglyLinkedListNode* node, string sep, ofstream& fout) {
    while (node) {
        fout << node->data;

        node = node->next;

        if (node) {
            fout << sep;
        }
    }
}

void free_singly_linked_list(SinglyLinkedListNode* node) {
    while (node) {
        SinglyLinkedListNode* temp = node;
        node = node->next;

        free(temp);
    }
}

// Complete the insertNodeAtPosition function below.

/*
 * For your reference:
 *
 * SinglyLinkedListNode {
 *     int data;
 *     SinglyLinkedListNode* next;
 * };
 *
 */
SinglyLinkedListNode* insertNodeAtPosition(SinglyLinkedListNode* head, int data, int position) {
     SinglyLinkedListNode* prev=new SinglyLinkedListNode(1);
     SinglyLinkedListNode* cur=new SinglyLinkedListNode(1);
     SinglyLinkedListNode* temp=new SinglyLinkedListNode(1);
    cur=head;
    for(int i=0;i<position;i++){
        prev=cur;
        cur=cur->next;
    }
temp->data=data;
    prev->next=temp;
    temp->next=cur;
    return head;
}

int main()
{
    ofstream fout(getenv("OUTPUT_PATH"));

    SinglyLinkedList* llist = new SinglyLinkedList();

    int llist_count;
    cin >> llist_count;
    cin.ignore(numeric_limits<streamsize>::max(), '\n');

    for (int i = 0; i < llist_count; i++) {
        int llist_item;
        cin >> llist_item;
        cin.ignore(numeric_limits<streamsize>::max(), '\n');

        llist->insert_node(llist_item);
    }

    int data;
    cin >> data;
    cin.ignore(numeric_limits<streamsize>::max(), '\n');

    int position;
    cin >> position;
    cin.ignore(numeric_limits<streamsize>::max(), '\n');

    SinglyLinkedListNode* llist_head = insertNodeAtPosition(llist->head, data, position);

    print_singly_linked_list(llist_head, " ", fout);
    fout << "\n";

    free_singly_linked_list(llist_head);

    fout.close();

    return 0;
}
```

Line: 27 Col: 1

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