Day 15: Linked List | HackerRank

hackerrank.com/challenges/30-linked-list/problem?h_r=internal-search&isFullScreen=true

HackerRank Prepare > Tutorials > 30 Days of Code > Day 15: Linked List

Exit Full Screen View

Problem | Submissions | Leaderboard | Discussions

Objective

Today we will work with a Linked List. Check out the Tutorial tab for learning materials and an instructional video.

A Node class is provided for you in the editor. A Node object has an integer data field, $data$, and a Node instance pointer, $next$, pointing to another node (i.e.: the next node in the list).

A Node insert function is also declared in your editor. It has two parameters: a pointer, $head$, pointing to the first node of a linked list, and an integer, $data$, that must be added to the end of the list as a new Node object.

Task

Complete the insert function in your editor so that it creates a new Node (pass $data$ as the Node constructor argument) and inserts it at the tail of the linked list referenced by the $head$ parameter. Once the new node is added, return the reference to the $head$ node.

Note: The $head$ argument is null for an empty list.

Input Format

The first line contains T, the number of elements to insert.
Each of the next T lines contains an integer to insert at the end of the list.

Output Format

Return a reference to the $head$ node of the linked list.

Sample Input

```
30
31 > void display(Node *head) ···
```

Line: 29 Col: 2

UploadCodeasFile | Test against custom input | Run Code | Submit Code

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin) Download

```
4
2
3
4
1
```

Your Output (stdout)

```
2 3 4 1
```

Expected Output Download

```
2 3 4 1
```

27°C Windy | Search | ENG IN | 09:27 21-07-2023

Day 15: Linked List | HackerRank

hackerrank.com/challenges/30-linked-list/problem?h_r=internal-search&isFullScreen=true

HackerRank Prepare > Tutorials > 30 Days of Code > Day 15: Linked List

Exit Full Screen View

Problem | Submissions | Leaderboard | Discussions

Objective

Today we will work with a Linked List. Check out the Tutorial tab for learning materials and an instructional video.

A Node class is provided for you in the editor. A Node object has an integer data field, $data$, and a Node instance pointer, $next$, pointing to another node (i.e.: the next node in the list).

A Node insert function is also declared in your editor. It has two parameters: a pointer, $head$, pointing to the first node of a linked list, and an integer, $data$, that must be added to the end of the list as a new Node object.

Task

Complete the insert function in your editor so that it creates a new Node (pass $data$ as the Node constructor argument) and inserts it at the tail of the linked list referenced by the $head$ parameter. Once the new node is added, return the reference to the $head$ node.

Note: The $head$ argument is null for an empty list.

Input Format

The first line contains T, the number of elements to insert.
Each of the next T lines contains an integer to insert at the end of the list.

Output Format

Return a reference to the $head$ node of the linked list.

Sample Input

Change Theme Language C

```c
#include <stdlib.h>···
Node* insert(Node *head,int data)
{
    //Complete this function
    Node *temp=(Node*)malloc(sizeof(Node));
        temp->data=data;
    temp->next=NULL;
    if(head==NULL)
    {
        head = temp;
        return head;
    }
    else
    {
        Node *start=head;
        while(start->next!=NULL)
        {
        start=start->next;
        }
        start->next=temp;
        return head;
    }
}

void display(Node *head)···
```

Line: 29 Col: 2

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

27°C Windy Search ENG IN 09:27 21-07-2023