A8-Q1

Due on 2019-09-28, 23:59 IST

In this question, you have to write code to remove the first node in a doubly linked list containing a specified number. The code to create the linked list is already given. The main() function calls a function removeData(head, tail, n) to remove the first node containing n from the linked list.

Complete the code by writing the missing function. You can also write additional functions which you may need.

Input Format

You will be given a positive number N in the first line.
This will be followed by a line containing N integers.
This will be followed by an integer M.

Output format

The output will be the list after deleting M. Note that you do not have to add any additional code to print the list - once you code the removeData() function, the code will print the contents of the remaining list.

```
#include <stdio.h>
#include <stdlib.h>

struct DoublyLinkedList{
    int data;
    struct DoublyLinkedList *prev;
};
```
struct DoublyLinkedList *next;
};

/*
--------------------------
Create a new node
--------------------------
*/
struct DoublyLinkedList *createNode(int n)
{
    struct DoublyLinkedList *newNodeptr;
    newNodeptr = (struct DoublyLinkedList*)malloc(sizeof(struct DoublyLinkedList));
    newNodeptr->data = n;
    newNodeptr->prev = NULL;
    newNodeptr->next = NULL;
    return newNodeptr;
}

/*
--------------------------------
add a node at the end of a doubly linked list.
Tailptr is the address of the pointer to the end of the current list.
After adding the node, tail points to the new node inserted.
--------------------------------
*/
void appendNode ( struct DoublyLinkedList **tailptr, int n )
{
    struct DoublyLinkedList *newNode;
    newNode = createNode ( n );
    newNode->prev=*tailptr;
    (*tailptr)->next = newNode;
    *tailptr=newNode;
}
void initializeList (  
    struct DoublyLinkedList **headptr,
    struct DoublyLinkedList **tailptr,
    int n)
{
    struct DoublyLinkedList *newNode;
    newNode=createNode(n);
    *headptr=newNode;
    *tailptr=newNode;
    return;
}

void printList(
    struct DoublyLinkedList *head,
    struct DoublyLinkedList *tail)
{
    struct DoublyLinkedList *curr=head;

    while ( curr != NULL ){
        if(curr->next!=NULL)
            printf("%d,",curr->data);
        else printf("%d",curr->data);
        curr = curr->next;
    }

    return;
}

/*
   --------------------------------
   remove the node that ptr points to.
   After removing the first node, we should reset head.
   After removing the last node, we should reset tail.
   --------------------------------
*/

int main()
{
    int n;
int i=0;
int m;
struct DoublyLinkedList *head, *tail;

scanf("%d",&n);

if ( n <= 0 ){
    return 0;
}

scanf("%d",&m);
initializeList(&head,&tail,m);

for(i=1;i<n;i++){ /* read the remaining elements */
    scanf("%d",&m);
    appendNode(&tail,m);
}

scanf("%d",&n);
removeData ( &head, &tail, n );
printList(head,tail);
return 0;

Sample Test Cases

<table>
<thead>
<tr>
<th>Test Case</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Case 1</td>
<td>6 1 2 3 4 5 6 1</td>
<td>2, 3, 4, 5, 6</td>
</tr>
<tr>
<td>Test Case 2</td>
<td>6 1 2 3 4 5 6 6</td>
<td>1, 2, 3, 4, 5</td>
</tr>
<tr>
<td>Test Case 3</td>
<td>6 1 2 3 4 5 6 3</td>
<td>1, 2, 4, 5, 6</td>
</tr>
</tbody>
</table>
Test Case 4

The due date for submitting this assignment has passed.
As per our records you have not submitted this assignment.

Sample solutions (Provided by instructor)

```c
#include <stdio.h>
#include <stdlib.h>

struct DoublyLinkedList{
    int data;
    struct DoublyLinkedList *prev;
    struct DoublyLinkedList *next;
};

/*
 * Create a new node
 */
struct DoublyLinkedList *createNode(int n)
{
    struct DoublyLinkedList *newNodeptr;
    newNodeptr = (struct DoublyLinkedList*)malloc(sizeof(struct DoublyLinkedList));
    newNodeptr->data = n;
    newNodeptr->prev = NULL;
    newNodeptr->next = NULL;
    return newNodeptr;
}

/*
 * Add a node at the end of a doubly linked list.
 * Tailptr is the address of the pointer to the end of the current list.
 * After adding the node, tail points to the new node inserted.
 */
void appendNode ( struct DoublyLinkedList **tailptr, int n )
{
    struct DoublyLinkedList *newNode;
    newNode = createNode ( n );
    newNode->prev=*tailptr;
    (*tailptr)->next = newNode;
    *tailptr=newNode;
}

void initializeList ( struct DoublyLinkedList **headptr,
                      struct DoublyLinkedList **tailptr,
                      int n)
{
    struct DoublyLinkedList *newNode;
    newNode=createNode(n);
    *headptr=newNode;
    *tailptr=newNode;
    return;
}

void printList(
    struct DoublyLinkedList *head,
    struct DoublyLinkedList *tail)
{
    struct DoublyLinkedList *curr=head;
    ```
while ( curr != NULL ){
    if(curr->next!=NULL)
        printf("%d,",curr->data);
    else printf("%d",curr->data);
    curr = curr->next;
}
return;
}

/*
 * remove the node that ptr points to.
 * After removing the first node, we should reset head.
 * After removing the last node, we should reset tail.
 * -----------------------------------------------
 */
void removeNode (struct DoublyLinkedList **headptr,
                 struct DoublyLinkedList **tailptr,
                 struct DoublyLinkedList *ptr)
{
    if ( ptr == *headptr ){
        *headptr = ptr->next;
        (*headptr)->prev = NULL;
    }else{
        if ( ptr == *tailptr ) {
            *tailptr = ptr->prev;
            (*tailptr)->next=NULL;
        }else{
            ptr->prev->next = ptr->next;
            ptr->next->prev = ptr->prev;
        }
    }free ( ptr );
    ptr = NULL;
return;
}

void removeData (struct DoublyLinkedList **headptr,
                 struct DoublyLinkedList **tailptr,
                 int n)
{
    struct DoublyLinkedList *curr;
    curr = *headptr;
    while ( curr != NULL && curr->data != n ){
        curr = curr->next;
    }
    if ( curr != NULL ){
        removeNode(headptr, tailptr, curr);
    }return;
}

int main()
{
    int n;
    int i=0;
    int m;
    struct DoublyLinkedList *head, *tail;
    scanf("%d",&n);
    if ( n <= 0 ){
        return 0;
    }
    scanf("%d",&m);
    initializeList(&head,&tail,m);
    for(i=1;i<n;i++) { /* read the remaining elements */
        scanf("%d",&m);
        appendNode(&tail,m);
    }
}
```c
149     scanf("%d",&n);
150     removeData ( &head, &tail, n );
151     printList(head,tail);
152
153  
154     return 0;
155 }
156
157
```