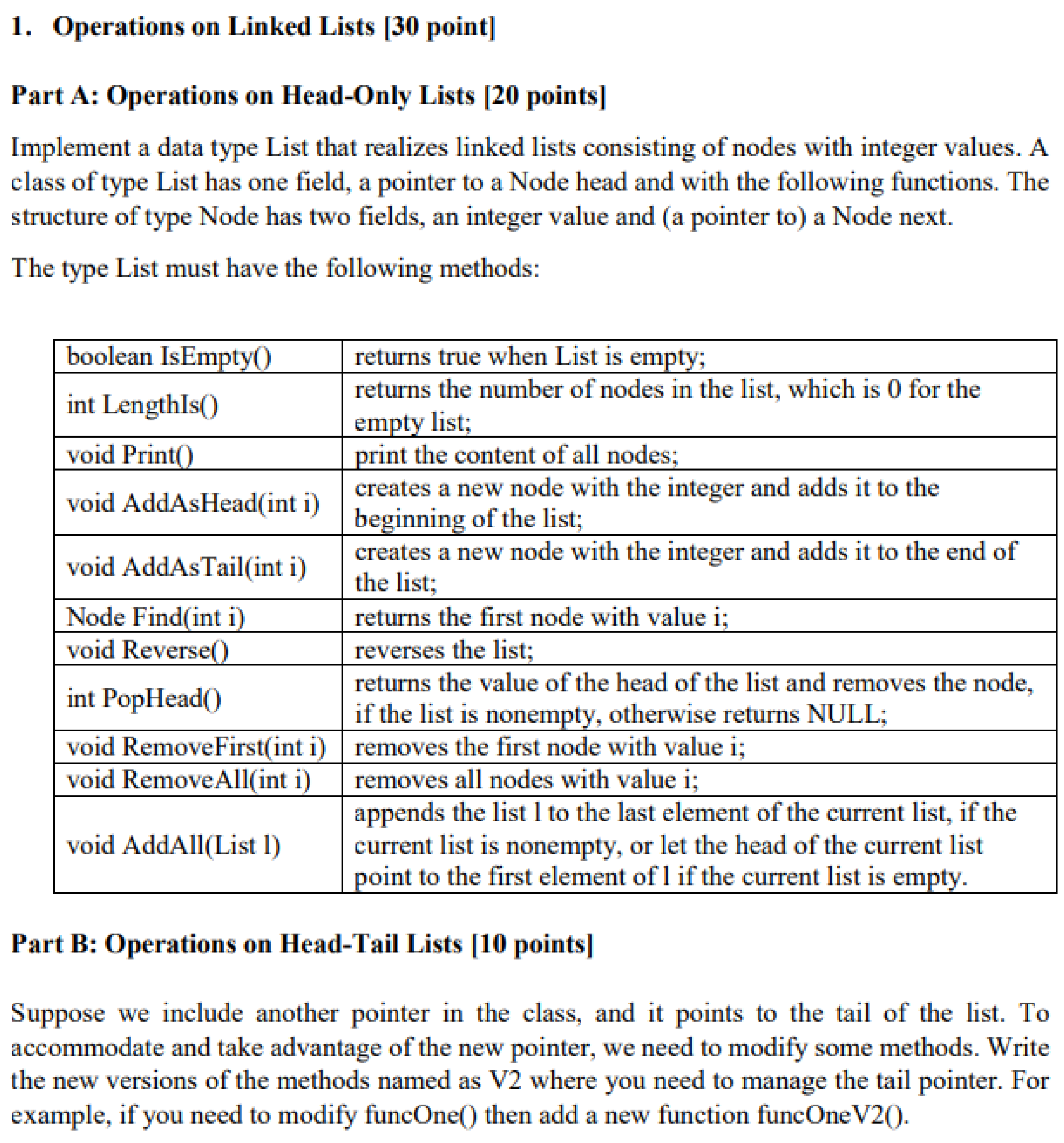
Operations on Linked Lists (30 point] Part A: Operations on Head-Only Lists [20 points] Implem...



1. **Operations on Linked Lists [30 point]**

Part A: Operations on Head-Only Lists [20 points]

Implement a data type List that realizes linked lists consisting of nodes with integer values. A class of type List has one field, a pointer to a Node head and with the following functions. The structure of type Node has two fields, an integer value and (a pointer to) a Node next.

The type List must have the following methods:

boolean IsEmpty()	returns true when List is empty;
int LengthIs()	returns the number of nodes in the list, which is 0 for the empty list;
void Print()	print the content of all nodes;
void AddAsHead(int i)	creates a new node with the integer and adds it to the beginning of the list;
void AddAsTail(int i)	creates a new node with the integer and adds it to the end of the list;
Node Find(int i)	returns the first node with value i;
void Reverse()	reverses the list;
int PopHead()	returns the value of the head of the list and removes the node, if the list is nonempty, otherwise returns NULL;
void RemoveFirst(int i)	removes the first node with value i;
void RemoveAll(int i)	removes all nodes with value i;
void AddAll(List l)	appends the list l to the last element of the current list, if the current list is nonempty, or let the head of the current list point to the first element of l if the current list is empty.

Part B: Operations on Head-Tail Lists [10 points]

Suppose we include another pointer in the class, and it points to the tail of the list. To accommodate and take advantage of the new pointer, we need to modify some methods. Write the new versions of the methods named as V2 where you need to manage the tail pointer. For example, if you need to modify funcOne() then add a new function funcOneV2().

Solve this question by using C++ Language

Show transcribed image text

Expert Answer

- Anonymous answered this
  Was this answer helpful?
  0
  0
  4 answers

```
#include<iostream>
#include<stdlib.h>
using namespace std;
//node structure
struct node{
int data;
struct node * next;
};
// function for print linkedlist
int linkedlist(struct node* ptr)
{
while(ptr!=NULL)
{
printf("%d->",ptr->data);
ptr=ptr->next;
}
cout<<"\n";
}
// function for insert value at first
struct node* insertatfirst(struct node * head, int data)
{
struct node * ptr=(struct node*)malloc(sizeof(struct node));
ptr->next=head;
ptr->data=data;
return ptr;
}
// function for insert value at any index
struct node* insertatindex(struct node * head, int data,int index)
{
struct node * ptr=(struct node*)malloc(sizeof(struct node*));
struct node * p = head;
int i=0;
while(i!=index-1)
{
p=p->next;
i++;
}
ptr->data=data;
ptr->next=p->next;
p->next=ptr;
```

```
return head;
}
//function for insert value at tail or last
struct node* insertatend(struct node * head, int data)
{
struct node * ptr=(struct node*)malloc(sizeof(struct node));
struct node * p = head;
ptr->data=data;
while(p->next!=NULL)
{
p=p->next;
}
p->next=ptr;
ptr->next=NULL;
return head;
}
//function for insert value after a particular node
struct node* insertafteranode(struct node * head, struct node* prenode, int data)
{
struct node* ptr= (struct node*)malloc(sizeof(struct node));
ptr->data=data;
ptr->next=prenode->next;
prenode->next=ptr;
return head;
}
//function for delete value at first or delete a head
struct node* deleteatfirst(struct node * head)
{
struct node* ptr=head;
head=ptr->next;
free(ptr);
return head;
}
//function for delete value from any index
struct node* deleteatindex(struct node * head, int index)
{
struct node* ptr=head ;
struct node * p = head->next;
for(int i=0;i<index-1;i++)
{
ptr=ptr->next;
p=p->next;
}
ptr->next=p->next;
free(p);
return head;
}
//function for delete value at end or tail
struct node* deleteatend(struct node * head)
{
```

```
struct node * p = head;
struct node * q = head->next;
while(q->next!=NULL)
{
p=p->next;
q=q->next;
}
p->next=NULL;
free(q);
return head;
}
//function for delete value by given any value which we want to delete
struct node* deleteanodefromvalue(struct node* head,int value)
{
struct node* ptr=head;
struct node* p=head->next;
while(p->data!=value)
{
ptr=ptr->next;
p=p->next;
}
ptr->next=p->next;
free(p);
return head;
}
int main()
{
struct node * head;
struct node * second;
struct node * third;
struct node * fourth;
head = (struct node *) malloc (sizeof (struct node));
second = (struct node *) malloc (sizeof (struct node));
third = (struct node *) malloc (sizeof (struct node));
fourth = (struct node *) malloc (sizeof (struct node));
head ->data = 8;
head -> next = second;
second ->data = 7;
second -> next = third;
third ->data = 9;
third -> next = fourth;
fourth ->data = 11;
fourth -> next = NULL;
linkedlist(head);
head=insertatfirst(head,56);
linkedlist(head);
head=insertatindex(head,90,4);
linkedlist(head);
head=insertatend(head,60);
linkedlist(head);
```

```
head=insertafteranode(head,third,567);
linkedlist(head);
head=deleteatfirst(head);
linkedlist(head);
head=deleteatindex(head,3);
linkedlist(head);
head=deleteatend(head);
linkedlist(head);
head=deleteanodefromvalue(head,90);
linkedlist(head);
return 0;
}
```