Alternating List Concatenation

locked

Complete the function 'concat'. It takes the head pointers of two lists as parameters. Concatenate the nodes in both the lists in an alternate manner to form a new list. Return the new list. The first element is from the list with head1. Note that if any one of the lists runs out of elements, the remaining elements of the other list must be appended. Make use of the pre-coded functions to finish the code. If both lists are empty, return NULL from the function.

Input Format

Number_of_elements_in_list1
Elements in list1 (if number of elements not 0)
Number_of_elements_in_list2
Elements in list2 (if number of elements not 0)

Constraints

The lists have at most 50 elements.

Output Format

Elements in the new list (space separated)

Sample Input 0

```
4
1 2 3 4
5
5 6 7 8 9
```

Sample Output 0

```
1 5 2 6 3 7 4 8 9
```

Explanation 0

Starting from the first element of the first list the numbers are alternatively added to a new list. Once the numbers in the first list are done the remaining numbers in the second list are added

Sample Input 1

```
3
4 7 1
0
```

Sample Output 1

```
4 7 1
```

Explanation 1

Since the second list is empty the final list is the first list itself.

Submissions: 62
Max Score: 10
Difficulty: Easy

Rate This Challenge:
☆☆☆☆☆

More

C

```
#include<stdio.h>
#include<stdlib.h>
typedef struct node
{
int info;
struct node* next;
}NODE;

NODE *createList(NODE *head,int ele);
NODE *getNode(int ele);
void display(NODE *head);
NODE *concat(NODE *head1,NODE *head2);
NODE *freeList(NODE *head);

int main()
{
NODE *head1=NULL;
NODE *head2=NULL;
int ele,n;
scanf("%d",&n);
for(int i=0;i<n;i++)
{
scanf("%d",&ele);
head1=createList(head1,ele);
}
scanf("%d",&n);
for(int i=0;i<n;i++)
{
scanf("%d",&ele);
head2=createList(head2,ele);
}
NODE *head3=concat(head1,head2);
display(head3);
freeList(head1);
freeList(head2);
freeList(head3);
return 0;

}


NODE *concat(NODE *head1, NODE *head2)
{
    NODE *head3 = NULL;
    NODE *p1 = head1;
    NODE *p2 = head2;

    while (p1 != NULL || p2 != NULL) {
        if (p1 != NULL) {
            head3 = createList(head3, p1->info);
            p1 = p1->next;
        }

        if (p2 != NULL) {
            head3 = createList(head3, p2->info);
            p2 = p2->next;
        }
```

```c
    }

    return head3;
}



NODE *getNode(int ele)
{
NODE *temp=malloc(sizeof(NODE));
temp->info=ele;
temp->next=NULL;
return temp;
}

NODE *createList(NODE *head,int ele)
{
NODE *temp=getNode(ele);
if(head==NULL)
head=temp;
else{
NODE *p=head;
while(p->next!=NULL)
p=p->next;
p->next=temp;}
return head;
}

void display(NODE *head)
{
NODE *p=head;
if(p==NULL)
{
printf("empty list\n");
}
else
{
while(p!=NULL)
{
printf("%d ",p->info);
p=p->next;
}
printf("\n");
}
}



NODE *freeList(NODE *head)
{
NODE *p=head;
NODE *q=NULL;
while(p!=NULL)
{
q=p;
p=p->next;
free(q);
}
head=NULL;
return head;
}
```

Line: 118 Col: 2

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