**数据结构习题（二）**

学号： 姓名：

**1、**3-22

#include <stdio.h>

#include <stdlib.h>

typedef struct node

{

char data;

struct node \*next;

}LinkList, Stack; // 链表和栈的结构体相同

int n;

LinkList \*Create() // 创建链表

{

char c;

int cnt = n;

LinkList \*head, \*s, \*r;

head = NULL; // 链表初值为空

r = NULL; // 尾指针初值为空

printf("请输入%d个数据\n", n);

while(cnt -- )

{

scanf("%c", &c);

s = (LinkList\*)malloc(sizeof(LinkList)); //分配空间

s -> data = c;

if(head == NULL)

head = s; // 新结点s插入空表

else

r -> next = s; //非空表，新结点s插入到链表尾

r = s; // 尾指针指向新表尾

}

if(r != NULL) //非空表，将尾结点的指针指向空

r -> next = NULL;

return head; // 返回指针头

}

Stack \*PushL(Stack \*S, char c) // 栈放入元素

{

Stack \*p;

p = (Stack\*)malloc(sizeof(Stack)); // 分配空间

p -> data = c;

p -> next = S; // 将新结点置于旧节点之前，实现先进后出

S = p; // 更新当前结点

return S;

}

Stack \*PopL(Stack \*S) // 栈弹出元素

{

if(S == NULL) // 当栈为空时，不可再弹出

{

printf("Stack is underflow\n");

return NULL;

}

Stack \*tmp = S;

S = S -> next; // 更新S，实现栈顶元素的弹出

free(tmp); // 释放栈顶的空间

return S;

}

int main()

{

printf("请输入一共存放的字符数：");

scanf("%d", &n);

getchar(); // 接收输入的回车

LinkList \*head = Create();

Stack \*top;

for(int i = 0; i < (n + 1) / 2; i ++ ) // 前一半元素（包括中间元素）入栈

{

char c = head -> data;

head = head -> next;

top = PushL(top, c);

}

int flag = 1; // 设置flag，判断是否对称

if(n % 2) top = PopL(top); // 当n为奇数时，弹出中间元素，因为不影响对称

for(LinkList \*i = head; i; i = i -> next) // 遍历链表剩余元素，并与栈中元素相比，得出是否对称

{

if(top -> data != i -> data)

{

flag = 0;

break;

}

top = PopL(top); // 弹出栈顶元素

}

if(flag)

printf("中心对称");

else

printf("非中心对称");

return 0;

}

**2、**3-26

队满条件：**quelen = m**;

#include <stdio.h>

#include <stdlib.h>

typedef int DataType; // 设置int为DataType

const int MAX = 10; // 队列最大长度设为10

typedef struct

{

DataType sequ[MAX];

int rear, quelen;

}sequeue; // 根据题意，设置sequ数组用来储存数据，rear记录队列末端位置，quelen用来记录队列长度

sequeue \*CreateQueue() // 创建队列

{

sequeue \*sq = (sequeue\*)malloc(sizeof(sequeue));

return sq;

}

sequeue \*SetNullQ(sequeue \*sq) // 队列置空

{

sq -> rear = MAX - 1;

sq -> quelen = 0;

return sq;

}

int EnQueueQ(sequeue \*sq, DataType data) // 入队

{

if(sq -> quelen >= MAX) // 当现有队列长度已经大于MAX时，说明队列已满，无法入队

{

printf("添加失败，队列已满！");

return 0;

}

sq -> rear = (sq -> rear + 1) % MAX; // 利用模运算避免讨论是否越界

sq -> sequ[sq -> rear] = data;

sq -> quelen ++ ; // 队列长度增加

return 1;

}

DataType DeQueueQ(sequeue \*sq) // 出队

{

if(sq -> quelen == 0) // 当quelen为0，即队列为空，无法出队

{

printf("出队失败，队伍为空！");

return 0;

}

DataType tmp; // 记录队首元素

sq -> quelen -- ; // 队列长度减1

tmp = sq -> sequ[(sq -> rear - sq -> quelen + MAX) % MAX]; // 利用模运算避免讨论

return tmp; // 返回队首元素

}

int main()

{

sequeue \*sq = CreateQueue();

sq = SetNullQ(sq);

// 以下为对代码的测试

for(int i = 0; i < 10; i ++ )

EnQueueQ(sq, i);

while(sq -> quelen != 0)

printf("%d\n", DeQueueQ(sq));

return 0;

}