**说明：大部分作业直接利用实验1第一题的链表操作linkedlist.c加以补充，减少代码重写率。剩下的第8题利用自己实现的stack.c。**

4. 参见附件linkedlist.c文件，以下是主要函数代码：

LinkedNode\* addTwoList(LinkedNode\* head1,LinkedNode\* head2){

LinkedNode \*head = create();

copy(head1,head);

LinkedNode \*p2 = head2;

while((p2 = p2->next)!=NULL){

insert(head,p2->element);

}

return head;

}

8. 参见附件stack.c文件，以下是主要函数代码：

int main()

{

char\* old\_str = "3\*-y-a/y!2"; // ! instand of

char\* new\_str = "3y-\*ay2!/-";

LinkedNode\* top = NULL;

while(\*new\_str){

top = push(top,\*old\_str);

old\_str++;

printf("X");

while(top && top->element == \*new\_str){

top = pop(top);

new\_str++;

printf("S");

}

}

return 0;

}

15. 环形表设计：头结点存链表长度length，正向查找与普通环形表相同，反向查找则每次向前前进length次。

顺时针访问链表函数参见附件linkedlist.c文件，以下是主要函数代码：

int find(LinkedNode\* head,ElementType x){

LinkedNode\* p = head;

while(p->next!=NULL && p->next!= head){

p = p->next;

if(p->element == x){

return TRUE;

}

}

return FALSE;

}

18. 参见附件linkedlist.c文件，以下是主要函数代码：

int findIndex(LinkedNode\* head,ElementType x){

LinkedNode\* p = head;

int index = 0;

while(p->next!=NULL){

p = p->next;

index ++;

if(p->element == x){

return index;

}

}

LinkedNode\* add\_list = (LinkedNode\*)malloc(sizeof(LinkedNode));

add\_list->element = x;

add\_list->next = NULL;

p->next = add\_list;

return -1;

}

19. 参见附件linkedlist.c文件，以下是主要函数代码：

void oneToTwo(LinkedNode\* head,LinkedNode\* head1,LinkedNode\* head2){

if(!head1) {head1 = create();}

if(!head2) {head2 = create();}

LinkedNode \*p = head;

int mark = 0;

while((p = p->next)!=NULL){

if(++mark%2){

insert(head1,p->element);

}else{

insert(head2,p->element);

}

}

makeCircle(head1);

head1->element = mark / 2 + 1;

makeCircle(head2);

head2->element = mark / 2;

}

22. 参见附件linkedlist.c文件，以下是主要函数代码：

ElementType\* subEle(LinkedNode\* head,int m,int n){

ElementType\* pe = (ElementType\* )malloc((n-m)\*sizeof(ElementType));

LinkedNode\* p = head;

int i,j;

for(i=0,j=0;p->next!=NULL;i++){

p = p->next;

if(i >= m && i < n){

pe[j] = p->element;

j++;

}

}

return pe;

}