**#include <stdio.h>**

**#include <conio.h>**

**#include<malloc.h>//#include "alloc.h"**

**typedef struct LNode{**

**int data;**

**struct LNode \*next;**

**}NODE;**

**NODE \*InitList() {**

**NODE \*head;**

**head = (NODE \*)malloc(sizeof(NODE));**

**head->next = NULL;**

**return head;**

**}**

**NODE \*OneCreate() {**

**NODE \*tail,\*head;**

**tail=head = InitList();**

**int n;**

**cout << "Please input the numbers if(-1) stop:";**

**cin >> n;**

**while (n !=-1) {**

**NODE \*p =(NODE\*)malloc(sizeof(NODE));**

**p->data = n;**

**p->next = NULL;**

**tail->next = p;**

**tail = p;**

**cout << "Please input the numbers if(-1) stop:";**

**cin >> n;**

**}**

**return head;**

**}**

**NODE \*TwoCreate() {**

**NODE \*head = InitList();**

**int n;**

**cout << "Please input the numbers if(-1) stop:\n";**

**cin >> n;**

**while (n != -1) {**

**NODE \*p;**

**p = new NODE;**

**p->data = n;**

**p->next = head->next;**

**head->next = p;**

**cout << "Please input the numbers if(-1) stop:\n";**

**cin >> n;**

**}**

**return head;**

**}**

**void print(NODE \*head) {**

**head = head->next;**

**while (head != NULL) {**

**cout <<setw(2)<< head->data;**

**head = head->next;**

**}**

**}**

**int DeleteList(NODE \*head, int num) {**

**NODE \*p, \*q;**

**p = head;**

**while (p->next != NULL && p->next->data != num) {**

**p = p->next;**

**}**

**q = p->next;**

**if (p->next != NULL) {**

**p->next = q->next;**

**free(q);**

**cout << "Deleted successfully\n";**

**}**

**else**

**cout << "Not found\n";**

**return 0;**

**}**

**int deletelist(NODE\*head, int i, int &e) {**

**NODE \*p = head;**

**int j = 1;**

**while (p != NULL && j < i) {**

**p = p->next;**

**j++;**

**}**

**if (p == NULL || j > i) return -1;**

**NODE \*q = p->next;**

**p->next = q->next;**

**e = q->data;**

**free(q);**

**return e;**

**}**

**int InsertList(NODE \*head, int i, int e) {**

**NODE \*p = head;**

**int j = 1;**

**while (p != NULL && j < i) {**

**p = p->next;**

**j++;**

**}**

**if (p == NULL || j > i) return false;**

**NODE \*s = (NODE\*)malloc(sizeof(NODE));**

**s->data = e;**

**s->next = p->next;**

**p->next = s;**

**return 0;**

**}**

**NODE \*ListSort(NODE \*head) {**

**for (NODE \*p1 = head->next; p1 != NULL; p1 = p1->next) {**

**for (NODE \*p2 = p1->next; p2 != NULL; p2 = p2->next) {**

**if (p2->data < p1->data) {**

**int temp = p2->data;**

**p2->data = p1->data;**

**p1->data = temp;**

**}**

**}**

**}**

**return head;**

**}**

**int insertlist(NODE \*head, int num) {**

**NODE \*p = head;**

**while (p!=NULL&&p->next->data < num) {**

**p = p->next;**

**}**

**NODE \*s = (NODE \*)malloc(sizeof(NODE));**

**s->data= num;**

**s->next = p->next;**

**p->next = s;**

**return 0;**

**}**

**int reverse(NODE \*head) {**

**NODE \* p1 = NULL;**

**NODE \*p2;**

**NODE \*p = head->next;**

**while (p != NULL) {**

**p2 = new NODE;**

**p2->next = p1;**

**p1 = p2;**

**p2->data = p->data;**

**p = p->next;**

**}**

**head->next = p1;**

**return 0;**

**}**

**NODE \* MergeList(NODE\*head1, NODE \*head2) {**

**NODE \*head3, \*p1, \*p2, \*p3;**

**p1 = head1->next;**

**p2 = head2->next;**

**head3 = (NODE \*)malloc(sizeof(NODE));**

**head3->next = NULL;**

**p3 = head3;**

**while(p1 &&p2) {**

**if (p1->data <= p2->data) {**

**p3->next = p1; p3 = p1; p1 = p1->next;**

**}**

**else {**

**p3->next = p2; p3 = p2; p2= p2->next;**

**}**

**}**

**p3->next = p1 ? p1 : p2;**

**free(head2);**

**return head3;**

**}**

**#include<iostream>**

**#include<iomanip>**

**using namespace std;**

**typedef struct LNode {**

**int data;**

**struct LNode \*next;**

**}NODE;**

**NODE \*InitList() {**

**NODE \*head;**

**head = (NODE \*)malloc(sizeof(NODE));**

**head->next = NULL;**

**return head;**

**}**

**NODE \*OneCreate() {**

**NODE \*tail, \*head;**

**tail = head = InitList();**

**int n;**

**cout << "Please input the numbers if(-1) stop:\n";**

**cin >> n;**

**while (n != -1) {**

**NODE \*p = new NODE;**

**p->data = n;**

**p->next = NULL;**

**tail->next = p;**

**tail = p;**

**cout << "Please input the numbers if(-1) stop:\n";**

**cin >> n;**

**}**

**return head;**

**}**

**NODE \*TwoCreate() {**

**NODE \*head = InitList();**

**int n;**

**cout << "Please input the numbers if(-1) stop:\n";**

**cin >> n;**

**while (n != -1) {**

**NODE \*p;**

**p = new NODE;**

**p->data = n;**

**p->next = head->next;**

**head->next = p;**

**cout << "Please input the numbers if(-1) stop:\n";**

**cin >> n;**

**}**

**return head;**

**}**

**int InsertList(NODE \*head, int num) {**

**NODE \*p = head;**

**while (p->next != NULL && p->next->data < num) {**

**p = p->next;**

**}**

**NODE \*s = (NODE \*)malloc(sizeof(NODE));**

**s->data = num;**

**s->next = p->next;**

**p->next = s;**

**return 0;**

**}**

**void print(NODE \*head) {**

**head = head->next;**

**while (head != NULL) {**

**cout << setw(2) << head->data;**

**head = head->next;**

**}**

**cout << endl;**

**}**

**NODE \*ListSort(NODE \*head) {**

**for (NODE \*p1 = head->next; p1 != NULL; p1 = p1->next) {**

**for (NODE \*p2 = p1->next; p2 != NULL; p2 = p2->next) {**

**if (p2->data < p1->data) {**

**int temp = p2->data;**

**p2->data = p1->data;**

**p1->data = temp;**

**}**

**}**

**}**

**return head;**

**}**

**NODE \* MergeList(NODE\*head1, NODE \*head2) {**

**NODE \*head3, \*p1, \*p2, \*p3;**

**p1 = head1->next;**

**p2 = head2->next;**

**head3 = (NODE \*)malloc(sizeof(NODE));**

**head3->next = NULL;**

**p3 = head3;**

**while (p1 &&p2) {**

**if (p1->data <= p2->data) {**

**p3->next = p1; p3 = p1; p1 = p1->next;**

**}**

**else {**

**p3->next = p2; p3 = p2; p2 = p2->next;**

**}**

**}**

**p3->next = p1 ? p1 : p2;**

**free(head2);**

**return head3;**

**}**

**int DeleteList(NODE \*head, int num) {**

**NODE \*p, \*q;**

**p = head;**

**while (p->next != NULL && p->next->data != num){**

**p = p->next;**

**}**

**q = p->next;**

**if (p->next != NULL) {**

**p->next = q->next;**

**free(q);**

**cout << "Deleted successfully\n";**

**}**

**else**

**cout << "Not found\n";**

**return 0;**

**}**

**int \*reverse(NODE \*head) {**

**NODE \* p1 = NULL;**

**NODE \*p2;**

**NODE \*p = head->next;**

**while (p != NULL) {**

**p2 = new NODE;**

**p2->next = p1;**

**p1 = p2;**

**p2->data = p->data;**

**p = p->next;**

**}**

**head->next = p1;**

**return 0;**

**}**

**int main()**

**{**

**NODE \*head1, \*head2, \*head3;**

**char c;**

**int num;**

**cout << "正序建立一个链表:\n";**

**head1 = OneCreate();**

**print(head1);**

**cout << "逆序建立一个链表:\n";**

**head2 = TwoCreate();**

**print(head2);**

**while (1) {**

**cout << "S:ListSort() D:DeleteList() I:InsertList() M:MergeListReversePrint E:exit()\n";**

**cin >> c;**

**switch (c) {**

**case's':ListSort(head1);**

**ListSort(head2);**

**cout <<;**

**break;**

**case'd':cout <<;**

**cin >> num;**

**DeleteList(head1, num);**

**break;**

**case'i':cout <<;**

**cin >> num;**

**InsertList(head2, num);**

**cout <;**

**break;**

**case'm':head3 = MergeList(head1, head2);**

**reverse(head3);**

**print(head3);**

**break;**

**case'e':exit(0);**

**break;**

**}**

**}**

**system("pause");**

**return 0;**

**}**