**#include<stdio.h>**

**#include<stdlib.h>**

**#define maxnumofstart 100**

**struct node**

**{**

**int data ;**

**struct node \*next ;**

**};**

**typedef struct node \* nodeptr ;**

**nodeptr getNode(int item)**

**{**

**nodeptr p ;**

**p=(nodeptr) malloc(sizeof(struct node));**

**p->data=item ;**

**p->next=NULL;**

**return p ;**

**}**

**nodeptr getStart(int z)**

**{**

**nodeptr start ;**

**start=(nodeptr)malloc(sizeof(struct node));**

**start->data=z;**

**start=NULL;**

**return start ;**

**}**

**/\***

**void create(nodeptr start[maxnumofstart],int j)**

**{**

**int z;**

**for(z=0;z<j;++)**

**{**

**nodeptr new\_node=NULL,current=NULL;**

**int n=0;**

**printf("How many Nodes to create at start : ");**

**scanf("%d",&n);**

**while(n>0)**

**{**

**new\_node=(nodeptr)malloc(sizeof(struct node));**

**printf("Enter the data : ");**

**scanf("%d",&new\_node->data);**

**new\_node->next=NULL;**

**if(start[z]==NULL)**

**{**

**start[z]=new\_node;**

**current=new\_node;**

**}**

**else**

**{**

**current->next=new\_node;**

**current=new\_node;**

**}**

**n--;**

**}**

**}**

**\*/**

**void display(nodeptr \*start)**

**{**

**nodeptr p;**

**if(\*start==NULL)**

**{**

**printf("is Empty\n");**

**return;**

**}**

**p=\*start;**

**while (p!=NULL)**

**{**

**printf("%d %p\n",p->data,p);**

**p=p->next;**

**}**

**}**

**void insertFirst(nodeptr \*start,int item)**

**{**

**nodeptr p=getNode(item);**

**p->next=\*start;**

**\*start=p;**

**}**

**void insertLast(nodeptr \*start,int item)**

**{**

**if (\*start==NULL)**

**insertFirst(start,item);**

**else**

**{**

**nodeptr p,q ;**

**p=getNode(item);**

**q=\*start ;**

**while(q->next!=NULL)**

**q=q->next;**

**q->next=p;**

**p->next=NULL;**

**}**

**}**

**void insertAfter(nodeptr \*start,int item,int pre)**

**{**

**nodeptr q,p=getNode(item);**

**q=\*start;**

**while(q->data!=pre && q!=NULL)**

**q=q->next;**

**if(q==NULL)**

**printf("Previous not found");**

**else**

**{**

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

**q->next=p ;**

**}**

**}**

**void insertSorted(nodeptr \*start,int item)**

**{**

**nodeptr p,r;**

**p=\*start;**

**r=NULL;**

**if(start==NULL)**

**insertFirst(start,item);**

**else**

**{**

**while(p->data<=item && p!=NULL)**

**{**

**r=p;**

**p=p->next;**

**}**

**if (r==NULL)**

**insertFirst(start,item);**

**else**

**{**

**insertAfter(start,item,r->data);**

**}**

**}**

**}**

**int deleteFirst(nodeptr \*start)**

**{**

**int x;**

**nodeptr p;**

**if(\*start==NULL)**

**{**

**printf("Linked List is Empty");**

**return -32768 ;**

**}**

**p=\*start;**

**\*start=p->next ;**

**x=p->data;**

**free(p);**

**return x;**

**}**

**int deleteAfter(nodeptr \*start,int pre)**

**{**

**int x;**

**nodeptr p,q;**

**q=\*start;**

**while(q->data!=pre && q!=NULL)**

**q=q->next;**

**if (\*start==NULL)**

**{**

**printf("Linked List is empty");**

**return -32768 ;**

**}**

**else if (q==NULL || q -> next==NULL)**

**{**

**printf("Previous is NULL or no node after previous to delete");**

**return -32768 ;**

**}**

**else**

**{**

**p=q->next;**

**x=p->data;**

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

**free(p);**

**return x ;**

**}**

**}**

**int deleteLast(nodeptr \*start)**

**{**

**int x;**

**nodeptr p,q;**

**p=\*start;**

**q=NULL;**

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

**{**

**q=p;**

**p=p->next;**

**}**

**if(p==NULL)**

**{**

**printf("Linked List is empty") ;**

**return -32768 ;**

**}**

**else if(q==NULL)**

**x=deleteFirst(start);**

**else**

**{**

**x=deleteAfter(start,q->data);**

**}**

**return x;**

**}**

**int deleteVX(nodeptr \*start,int item)**

**{**

**int x;**

**nodeptr p,q;**

**p=\*start;**

**q=NULL;**

**while(p->data!=item && p!=NULL)**

**{**

**q=p ;**

**p=p->next;**

**}**

**if (p==NULL)**

**{**

**if(q==NULL)**

**printf("Linked List is Empty");**

**else**

**printf("Item not Found");**

**return -32768 ;**

**}**

**else**

**{**

**if (q==NULL)**

**x=deleteFirst(start);**

**else**

**{**

**x=deleteAfter(start,q->data);**

**}**

**return x;**

**}**

**}**

**void copy(nodeptr \*from,nodeptr \*to)**

**{**

**nodeptr p=\*to,q,r;**

**int n=0;**

**/\*while(p!=NULL)**

**{**

**q=p;**

**p=p->next;**

**free (q);**

**}\*/**

**p=\*from;**

**while(p!=NULL)**

**{**

**n++;**

**p=p->next;**

**}**

**q=\*from;**

**\*to=p=getNode(q->data);**

**n--;**

**while(n!=0)**

**{**

**q=q->next;**

**r=p;**

**p=getNode(q->data);**

**r->next=p;**

**n--;**

**}**

**p->next=NULL;**

**}**

**void concat(nodeptr \*from,nodeptr \*to)**

**{**

**nodeptr p=\*to,q=NULL,r;**

**while(p!=NULL)**

**{**

**q=p;**

**p=p->next;**

**}**

**p=\*from;**

**while(p!=NULL)**

**{**

**r=getNode(p->data);**

**q->next=r;**

**q=q->next;**

**p=p->next;**

**}**

**q->next=NULL;**

**}**

**void reverse(nodeptr \*start)**

**{**

**nodeptr prev = NULL;**

**nodeptr current = \*start;**

**nodeptr next;**

**while (current != NULL)**

**{**

**next = current->next;**

**current->next = prev;**

**prev = current;**

**current = next;**

**}**

**\*start = prev;**

**}**

**void split(int x,nodeptr start[maxnumofstart],int y,int \*i)**

**{**

**int counter;**

**nodeptr p=start[x];**

**while(y!=1 && p->next!=NULL)**

**{**

**p=p->next;**

**y--;**

**}**

**if (p==NULL || p->next==NULL)**

**return;**

**for(counter=\*i;counter>x+1;counter--)**

**{**

**start[counter]=start[counter-1];**

**}**

**start[counter]=p->next;**

**p->next=NULL;**

**\*i=\*i+1;**

**}**

**/\***

**void overflow(int \*i,int \*j)**

**{**

**while(\*j>=\*i)**

**{**

**\*i=\*j+1;**

**}**

**}**

**\*/**

**void reduce(int x,nodeptr start[maxnumofstart],int \*i)**

**{**

**for (;x<\*i-1;x++)**

**{**

**start[x]=start[x+1];**

**}**

**\*i=\*i-1;**

**}**

**void createList(nodeptr \*start,int \*i)**

**{**

**if(\*i<maxnumofstart)**

**{**

**\*start=getStart(\*i);**

**\*i=\*i+1;**

**}**

**}**

**void deleteList(nodeptr start[maxnumofstart],int \*i,int x)**

**{**

**int z;**

**for(z=x;z<\*i-1;z++)**

**{**

**start[z]=start[z+1];**

**}**

**\*i=\*i-1;**

**}**

**nodeptr sortList(nodeptr start)**

**{**

**nodeptr prev=NULL;**

**//**

**if(start == NULL || start->next == NULL)**

**return start; // the List is sorted.**

**//replace largest node with the first :**

**//1- find largest node :**

**nodeptr curr, largest,largestPrev;**

**curr = start;**

**largest = start;**

**prev = start;**

**largestPrev = start;**

**while(curr != NULL)**

**{**

**if(curr->data > largest->data)**

**{**

**largestPrev = prev;**

**largest = curr;**

**}**

**prev = curr;**

**curr = curr->next;**

**}**

**//largest node is in largest.**

**//2- switching firt node and largest node :**

**nodeptr tmp;**

**if(largest != start)**

**{**

**largestPrev->next = start;**

**tmp = start->next;**

**start->next = largest->next;**

**largest->next = tmp;**

**}**

**// now largest is the first node of the List.**

**// calling the function again with the sub List :**

**// \*start minus its first node :**

**largest->next = sortList(largest->next);**

**return largest;**

**}**

**void swapList(nodeptr \*s1,nodeptr \*s2)**

**{**

**nodeptr t1=\*s1,t2=\*s2;**

**\*s1=t2;**

**\*s2=t1;**

**}**

**void search(nodeptr start[maxnumofstart],int i,int item)**

**{**

**int z=0;**

**int c;**

**int notfound=0;**

**nodeptr p;**

**while(z!=i)**

**{**

**c=1;**

**p=start[z];**

**while(p!=NULL)**

**{**

**if(p->data==item)**

**{**

**notfound=1;**

**printf("Element Found in List %d at Location %d\n",(z+1),c);**

**}**

**p=p->next;**

**c++;**

**}**

**z++;**

**}**

**if (notfound==0)**

**{**

**printf("Element Not Found");**

**}**

**fflush(stdin);**

**getchar();**

**}**

**void main()**

**{**

**int n,x,y,z,c=1,i,j,no ;**

**printf("Enter the no of Linked List to work with : ");**

**scanf("%d",&i);**

**if (i<=0)**

**{**

**printf("Minimum of 1 Linked List must be created\nPress any key to continue .......");**

**fflush(stdin);**

**getchar();**

**i=1;**

**}**

**else if(i>100)**

**{**

**printf("Maximum Lists Allowed is 100\nPress any key to continue .......");**

**fflush(stdin);**

**getchar();**

**i=100;**

**}**

**nodeptr start[maxnumofstart];**

**for(z=0;z<i;z++)**

**{**

**start[z]=getStart(z);**

**}**

**/\* Doesn't Work**

**printf("How many lists to create Initially ? : ");**

**scanf("%d",&j);**

**if(j>0 && j<101)**

**{**

**create(start,j);**

**}**

**else if(j<=0)**

**{**

**create(start,1);**

**}**

**\*/**

**do**

**{**

**system("cls");**

**for(z=0;z<i;z++)**

**{**

**if (start[z]==NULL)**

**{**

**printf("List %d ",(z+1));**

**}**

**else**

**{**

**printf("List %d\n",(z+1));**

**}**

**display(&start[z]);**

**}**

**printf("\n1.Insert First\n2.Insert Last\n3.Insert After\n4.Insert Sorted\n5.Delete First\n6.Delete Last\n7.Delete After\n8.Delete With Value X\n9.Copy\n10.Concat\n11.Reverse\n12.Split\n13.Create List\n14.Delete List\n15.Sort\n16.Swap List\n17.Search\n");**

**scanf("%d",&n) ;**

**if ((n>0 && n<5) || n==8)**

**{**

**printf("Insert item ");**

**scanf("%d",&x) ;**

**}**

**if (n==3 || n==7)**

**{**

**printf("Enter Previous Value ");**

**scanf("%d",&y);**

**}**

**if(n>0 && n<9 && i!=1)**

**{**

**printf("Which List to work on : ");**

**scanf("%d",&c);**

**}**

**if (c>0 && c<=z)**

**{**

**no=c-1;**

**}**

**else**

**{**

**printf("\nWrong Choice\nPress Any Key to Continue .......");**

**fflush(stdin) ;**

**getchar();**

**c=1;**

**continue;**

**}**

**switch(n)**

**{**

**case 1 :insertFirst(&start[no],x);**

**break;**

**case 2 :insertLast(&start[no],x);**

**break;**

**case 3 :insertAfter(&start[no],x,y);**

**break;**

**case 4 :insertSorted(&start[no],x);**

**break;**

**case 5 :deleteFirst(&start[no]);**

**break;**

**case 6 :deleteLast(&start[no]);**

**break;**

**case 7 :deleteAfter(&start[no],y);**

**break;**

**case 8 :deleteVX(&start[no],x);**

**break;**

**case 9 :if(i==1)**

**{**

**printf("Copy not Possible with One List\nPress Any Key to Continue .......");**

**fflush(stdin);**

**getchar();**

**break;**

**}**

**printf("\nCopy list X to Y ");**

**scanf("%d %d",&x,&y);**

**x--;y--;**

**if(x<i && y<i && x>=0 && y>=0)**

**{**

**copy(&start[x],&start[y]);**

**}**

**else**

**{**

**printf("Wrong Input\nPress Any Key to Continue .......");**

**fflush(stdin);**

**getchar();**

**}**

**break;**

**case 10:if(i==1)**

**{**

**printf("Concat not Possible with One List\nPress Any Key to Continue .......");**

**getchar();**

**break;**

**}**

**printf("Concat list X at end of Y ");**

**scanf("%d %d",&x,&y);**

**x--;y--;**

**if(x<i && y<i && x>=0 && y>=0)**

**{**

**concat(&start[x],&start[y]);**

**}**

**else**

**{**

**printf("Wrong Input\nPress Any Key to Continue .......");**

**fflush(stdin);**

**getchar();**

**}**

**reduce(x,start,&i);**

**break;**

**case 11:if(i==1)**

**{**

**x=1;**

**}**

**else**

**{**

**printf("Reverse Which List : ");**

**scanf("%d",&x);**

**}**

**x--;**

**if(x>=0 && x<i)**

**{**

**reverse(&start[x]);**

**}**

**else**

**{**

**printf("Wrong Input\nPress Any Key to Continue .......");**

**fflush(stdin);**

**getchar();**

**}**

**break;**

**case 12:if(i==1)**

**{**

**x=1;**

**}**

**else if (i>1)**

**{**

**printf("Split Which List : ");**

**scanf("%d",&x);**

**}**

**printf("Which Position After Which to Start New List : ");**

**scanf("%d",&y);**

**x--;**

**if(x>=0 && x<i)**

**{**

**split(x,start,y,&i);**

**}**

**else**

**{**

**printf("Wrong Input\nPress Any Key to Continue .......");**

**fflush(stdin);**

**getchar();**

**}**

**break;**

**case 13:createList(&start[i],&i);**

**break;**

**case 14:if(i==1)**

**{**

**x=1;**

**createList(&start[i],&i);**

**}**

**else**

**{**

**printf("Delete Which List : ");**

**scanf("%d",&x);**

**}**

**x--;**

**if(x>=0 && x<i)**

**{**

**deleteList(start,&i,x);**

**}**

**else**

**{**

**printf("Wrong Input\nPress Any Key to Continue .......");**

**fflush(stdin);**

**getchar();**

**}**

**break;**

**case 15:if(i==1)**

**{**

**x=1;**

**}**

**else**

**{**

**printf("Which List to Sort : ");**

**scanf("%d",&x);**

**}**

**x--;**

**if(x>=0 && x<i)**

**{**

**start[x]=sortList(start[x]);**

**}**

**else**

**{**

**printf("Wrong Input\nPress Any Key to Continue .......");**

**fflush(stdin);**

**getchar();**

**}**

**break;**

**case 16:if(i==2)**

**{**

**x=1;**

**y=2;**

**}**

**else**

**{**

**printf("Which Lists to Swap : ");**

**scanf("%d %d",&x,&y);**

**}**

**x--;y--;**

**if(x<i && y<i && x>=0 && y>=0)**

**{**

**swapList(&start[x],&start[y]);**

**}**

**else**

**{**

**printf("Wrong Input\nPress Any Key to Continue .......");**

**fflush(stdin);**

**getchar();**

**}**

**break;**

**case 17:printf("Enter Element to Search : ");**

**scanf("%d",&x);**

**search(start,i,x);**

**break;**

**}**

**}while(n>0 && n<18) ;**

**}**











