#include <stdio.h>

#include<stdlib.h>

struct node

{

int info;

struct node \*link;

};

typedef struct node \*NODE;

NODE getnode()

{

NODE x;

x=(NODE)malloc(sizeof(struct node));

if(x==NULL)

{

printf("\n out of memory\n");

exit(0);

}

return x;

}

NODE insert\_front(int item,NODE first)

{

NODE temp;

temp=getnode();

temp->info=item;

temp->link=first;

return temp;

}

NODE reverse(NODE first)

{

NODE cur,temp;

cur=NULL;

while(first!=NULL)

{

temp=first;

first=first->link;

temp->link=cur;

cur=temp;

}

return cur;

}

int length(NODE first)

{

NODE cur;

int count=0;

cur=first;

while(cur!=NULL)

{

count++;

cur=cur->link;

}

return count;

}

void display(NODE first)

{

NODE cur;

if(first==NULL)

{

printf("list is empty\n");

return;

}

printf("the contents of the list are\n");

cur=first;

while(cur!=NULL)

{

printf("%d\n",cur->info);

cur=cur->link;

}

printf("\n");

}

void main()

{

NODE first;

int ch ,item;

first=NULL;

for(;;)

{

printf("enter your choice\n");

scanf("%d",&ch);

switch(ch)

{

case 1:printf("enter the item to be inserted\n");

scanf("%d",&item);

first=insert\_front(item,first);

break;

case 2:

printf("length=%d\n",length(first));

break;

case 3:first=reverse(first);

display(first);

break;

case 4:

display(first);

}

}

}

/\* implementation of queues by linked list \*/

#include<stdio.h>

#include<stdlib.h>

struct node

{

int info;

struct node \*link;

};

struct node \*front;

struct node \*rear;

typedef struct node \*NODE;

void main ()

{

int choice;

while(choice != 4)

{

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Main Menu\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\n=================================================================\n");

printf("\n1.insert an element\n2.Delete an element\n3.Display the queue\n4.Exit\n");

printf("\nEnter your choice: ");

scanf("%d",& choice);

switch(choice)

{

case 1:

insert();

break;

case 2:

delete();

break;

case 3:

display();

break;

case 4:

exit(0);

break;

default:

printf("\nEnter valid choice: \n");

}

}

}

void insert()

{

NODE ptr;

int item;

ptr = (NODE) malloc (sizeof(struct node));

if(ptr == NULL)

{

printf("\nOVERFLOW\n");

return;

}

else

{

printf("\nEnter value: \n");

scanf("%d",&item);

ptr -> info = item;

if(front == NULL)

{

front = ptr;

rear = ptr;

front -> link = NULL;

rear -> link = NULL;

}

else

{

rear -> link = ptr;

rear = ptr;

rear-> link = NULL;

}

}

}

void delete ()

{

NODE ptr;

if(front == NULL)

{

printf("\nUNDERFLOW\n");

return;

}

else

{

ptr = front;

front = front -> link;

free(ptr);

}

}

void display()

{

NODE ptr;

ptr = front;

if(front == NULL)

{

printf("\nEmpty queue\n");

}

else

{ printf("\nThe values of the queue are\n");

while(ptr != NULL)

{

printf("\n%d\n",ptr -> info);

ptr = ptr -> link;

}

}

}

/\* Implementation of stacks by linked list \*/

#include <stdio.h>

#include <stdlib.h>

struct node

{

int info;

struct node \*link;

};

struct node \*head;

typedef struct node \*NODE;

void main ()

{

int choice=0;

printf("\n\*\*\*\*\*\*\*\*\*Stack operations using linked list\*\*\*\*\*\*\*\*\*\n");

printf("\n----------------------------------------------\n");

while(choice != 4)

{

printf("\n\nChose one from the below options...\n");

printf("\n1.Push\n2.Pop\n3.display\n4.Exit");

printf("\n Enter your choice \n");

scanf("%d",&choice);

switch(choice)

{

case 1:

{

push();

break;

}

case 2:

{

pop();

break;

}

case 3:

{

display();

break;

}

case 4:

{

printf("Exiting....");

break;

}

default:

{

printf("Please Enter valid choice ");

}

}

}

}

void push ()

{

int info;

NODE ptr;

ptr = (NODE)malloc(sizeof(struct node));

if(ptr == NULL)

{

printf("not able to push the element");

}

else

{

printf("Enter the value");

scanf("%d",&info);

if(head==NULL)

{

ptr->info = info;

ptr -> link = NULL;

head=ptr;

}

else

{

ptr->info = info;

ptr->link = head;

head=ptr;

}

printf("Item pushed");

}

}

void pop()

{

int item;

NODE ptr;

if (head == NULL)

{

printf("Underflow");

}

else

{

item = head->info;

ptr = head;

head = head->link;

free(ptr);

printf("Item popped");

}

}

void display()

{

int i;

NODE ptr;

ptr=head;

if(ptr == NULL)

{

printf("Stack is empty\n");

}

else

{

printf("Printing Stack elements \n");

while(ptr!=NULL)

{

printf("%d\n",ptr->info);

ptr = ptr->link;

}

}

}

/\* Union and Intersection \*/

#include<stdio.h>

#include<stdlib.h>

struct node

{

int info;

struct node \*link;

};

typedef struct node \*NODE;

int search(int key, NODE first)

{

NODE cur;

if(first==NULL)

return 0;

cur=first;

while(cur!=NULL)

{

if(key==cur->info)

return 1;

cur=cur->link;

}

return 0;

}

void display(NODE first)

{

NODE cur;

if(first==NULL)

{

printf("List is empty\n");

return;

}

printf("The contents of singly linked list is\n");

cur=first;

while(cur!=NULL)

{

printf("%d\n",cur->info);

cur=cur->link;

}

printf("\n");

}

NODE getnode()

{

NODE x;

x=(NODE) malloc(sizeof(struct node));

if(x==NULL)

{

printf("Out of memory\n");

exit(0);

}

return x;

}

NODE insert\_rear(int item,NODE first)

{

NODE temp;

NODE cur;

temp=getnode();

temp->info=item;

temp->link=NULL;

if(first==NULL)

return temp;

cur=first;

while(cur->link!=NULL)

{

cur=cur->link;

}

cur->link=temp;

return first;

}

NODE remove\_duplicate(NODE first)

{

NODE a, b;

int flag;

if(first==NULL)

return NULL;

b=NULL;

a=first;

while(a!=NULL)

{

flag=search(a->info,b);

if(flag==0)

b=insert\_rear(a->info,b);

a=a->link;

}

return b;

}

NODE union\_of\_list(NODE first, NODE second)

{

NODE a, third;

int flag;

a=first;

third=NULL;

while(a!=NULL)

{

third=insert\_rear(a->info,third);

a=a->link;

}

a=second;

while(a!=NULL)

{

flag=search(a->info,third);

if(flag==0)

third=insert\_rear(a->info,third);

a=a->link;

}

return third;

}

NODE intersection\_of\_list(NODE first, NODE second)

{

NODE a,b,third;

int flag;

a=first;

b=second;

third=NULL;

while(a!=NULL)

{

flag=search(a->info,b);

if(flag==1)

third=insert\_rear(a->info,third);

a=a->link;

}

return third;

}

void main()

{

NODE first,second,third;

int choice,item,i,n;

for(;;)

{

printf("1:Create first list\n");

printf("2:Create second list\n");

printf("3:Remove duplicates of list 1\n");

printf("4:Remove duplicates of list 2\n");

printf("5:Union of two lists\n");

printf("6: Intersection of two lists\n");

printf("7:Exit\n");

printf("Enter the choice\n");

scanf("%d",& choice);

switch(choice)

{

case 1:

printf("enter the number of nodes in the LIST 1\n");

scanf("%d",&n);

first=NULL;

for(i=1;i<=n;i++)

{

printf("enter the item\n");

scanf("%d",&item);

first=insert\_rear(item,first);

}

break;

case 2:

printf("Enter the number of nodes in second list\n");

scanf("%d",&n);

second=NULL;

for(i=1;i<=n;i++)

{

printf("enter the item\n");

scanf("%d",&item);

second=insert\_rear(item,second);

}

break;

case 3:

printf("The first list before removing duplicate is:\n");

display(first);

first=remove\_duplicate(first);

printf("The first list after removing duplicates\n");

display(first);

break;

case 4:

printf("The second list before removing duplicate is:\n");

display(second);

second=remove\_duplicate(second);

printf("The first list after removing duplicates\n");

display(second);

break;

case 5:

printf("The first list is \n");

display(first);

printf("The second list is \n");

display(second);

third=union\_of\_list(first,second);

printf("The union of two lists\n");

display(third);

break;

case 6:

printf("The first list \n");

display(first);

printf("The second list\n");

display(second);

third=intersection\_of\_list(first,second);

printf("The intersection of two lists \n");

display(third);

break;

default:exit(0);

}

}

}