#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node\*next;

};

struct node\*head=NULL;

int isEmpty()

{

if (head==NULL)

return 1;

else

return 0;

}

void insertionAtBeginning()

{

struct node\*newnode=(struct node\*)malloc(sizeof(struct node));

int value;

printf("Enter the element to be inserted: ");

scanf("%d",&value);

newnode->data=value;

newnode->next=head;

head=newnode;

printf("Succesfully inserted the element at the beginning\n");

}

void insertionAtEnd()

{

if(isEmpty())

insertionAtBeginning();

else

{

struct node\*newnode=(struct node\*)malloc(sizeof(struct node));

int value;

printf("Enter the element to be inserted: ");

scanf("%d",&value);

struct node\*temp=head;

while(temp->next!=NULL)

{

temp=temp->next;

}

newnode->data=value;

temp->next=newnode;

newnode->next=NULL;

printf("Succesfully inserted the element at the end\n");

}

}

void insertionAtSpecificPosition()

{

struct node\*newnode=(struct node\*)malloc(sizeof(struct node));

int value;

printf("Enter the element to be inserted: ");

scanf("%d",&value);

int pos;

printf("Enter the postion where the element to be inserted: ");

scanf("%d",&pos);

if(pos<1)

printf("Invalid Position\n");

else if(pos==1)

insertionAtBeginning();

else

{

struct node\*temp=head;

for(int i=1;i<pos-1 && temp!=NULL;i++)

{

temp=temp->next;

}

newnode->data=value;

newnode->next=temp->next;

temp->next=newnode;

}

printf("Successfully inserted the element at the specified position\n");

}

void deleteFromBeginning()

{

if(isEmpty())

{

printf("List is Empty\n");

}

else

{

struct node\*temp=head;

head=temp->next;

free(temp);

printf("Successfully deleted the element from the beginning\n");

}

}

void deleteFromEnd()

{

if(isEmpty())

printf("List is Empty\n");

else

{

struct node\*temp=head;

struct node\*prev=NULL;

while(temp->next!=NULL)

{

prev=temp;

temp=temp->next;

}

prev->next=NULL;

free(temp);

printf("Successfully deleted the element from the end\n");

}

}

void deleteFromSpecificPosition()

{

if(isEmpty())

printf("List is Empty\n");

else

{

struct node\*temp=head;

int pos;

printf("Enter the position from which the element to be deleted: ");

scanf("%d",&pos);

if (pos<1)

printf("Invalid position\n");

else if(pos==1)

deleteFromBeginning();

else

{

struct node\*prev=NULL;

for(int i=1;i<pos && temp!=NULL;i++)

{

prev=temp;

temp=temp->next;

}

prev->next=temp->next;

free(temp);

printf("Successfully deleted the element at the specified position\n");

}

}

}

void display()

{

if(isEmpty())

printf("List is Empty\n");

else

{

struct node\*temp=head;

printf("The elements are:\n");

while(temp!=NULL)

{

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

temp=temp->next;

}

}

}

void search()

{

if(isEmpty())

printf("List is Empty\n");

else

{

struct node\*temp=head;

int value;

printf("Enter the value to be searched: ");

scanf("%d",&value);

int found=0;

while(temp)

{

if(temp->data==value)

{

found=1;

break;

}

temp=temp->next;

}

if(found==1)

{

printf("Element Found\n");

}

else

{

printf("Element Not Found\n");

}

}

}

int main()

{

int choice;

while(1)

{

printf("\n1.Insertion at beginning\n2.Insertion at End\n3.Insertion at a specific position\n");

printf("4.Deletion from beginning\n5.Deletion from end\n6.Deletion from a specific position\n");

printf("7.Display\n8.Search\n9.Exit\nEnter your choice: ");

scanf("%d",&choice);

switch(choice)

{

case 1:

insertionAtBeginning();

break;

case 2:

insertionAtEnd();

break;

case 3:

insertionAtSpecificPosition();

break;

case 4:

deleteFromBeginning();

break;

case 5:

deleteFromEnd();

break;

case 6:

deleteFromSpecificPosition();

break;

case 7:

display();

break;

case 8:

search();

break;

case 9:

exit(0);

break;

default:

printf("Invalid Choice\n");

break;

}

}

}