#include <stdio.h>

#include <stdlib.h>

struct node {

struct node\* prev;

int data;

struct node\* next;

};

struct node \*head=NULL;

//void createList();

void traverse();

void insertAtFront();

void insertAtEnd();

void insertAtPosition();

void deleteFirst();

void deleteEnd();

void deletePosition();

int main()

{

int choice;

while (1) {

//printf("\n\t1 Create List\n");

printf("\t1 To insert at beginning\n");

printf("\t2 To insert at end\n");

printf("\t3 To insert at middle\n");

printf("\t4 To delete from beginning\n");

printf("\t5 To delete from end\n");

printf("\t6 To delete at a poistion\n");

printf("\t7 To see list\n");

printf("\t8 To exit\n");

printf("Enter Choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

insertAtFront();

break;

case 2:

insertAtEnd();

break;

case 3:

insertAtPosition();

break;

case 4:

deleteFirst();

break;

case 5:

deleteEnd();

break;

case 6:

deletePosition();

break;

case 7:

traverse();

break;

case 8:

exit(1);

}

}

}

// Function to traverse the linked list

void traverse()

{

struct node \*temp;

// List is empty

if (head == NULL)

printf("\nList is empty\n");

else {

temp = head;

while (temp != NULL)

{

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

temp = temp->next;

}

}

}

void insertAtFront()

{

int data;

struct node\* temp, \*new;

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

printf("\nEnter number to be inserted: ");

scanf("%d", &data);

new->data = data;

if(head==NULL)

{

new->next = NULL;

new->prev =NULL;

head=new;

}

else

{

new->prev =NULL;

new->next = head;

head->prev= new;

head = new;

}

}

void insertAtEnd()

{

int data;

struct node \*temp, \*trav,\*newnode;

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

printf("\nEnter number to be inserted: ");

scanf("%d", &data);

newnode->data = data;

if(head==NULL)

{

newnode->next=NULL;

newnode->prev=NULL;

head=newnode;

}

temp = head;

while (temp->next != NULL)

{

temp = temp->next;

}

newnode->prev = temp;

temp->next = newnode;

}

void insertAtPosition()

{

int data, pos, i = 1;

struct node \*temp, \*newnode;

newnode = malloc(sizeof(struct node));

newnode->next = NULL;

newnode->prev = NULL;

printf("\nEnter position : ");

scanf("%d", &pos);

if (pos == 1)

{

newnode->next = head;

newnode->next->prev = newnode;

newnode->prev = NULL;

head = newnode;

}

// Change links

else

{

printf("\nEnter number to be inserted: ");

scanf("%d", &data);

newnode->data = data;

temp = head;

while (i < pos - 1)

{

temp = temp->next;

i++;

}

newnode->next = temp->next;

newnode->prev = temp;

temp->next = newnode;

temp->next->prev = newnode;

}

}

void deleteFirst()

{

struct node\* temp;

if (head== NULL)

printf("\nList is empty\n");

else {

temp = head;

head = head->next;

head->prev = NULL;

free(temp);

}

}

void deleteEnd()

{

struct node\* temp;

if (head == NULL)

printf("\nList is empty\n");

temp = head;

while (temp->next != NULL)

temp = temp->next;

if (head->next == NULL)

head = NULL;

else {

temp->prev->next = NULL;

free(temp);

}

}

void deletePosition()

{ struct node \*temp, \*prev, \*ptr;

int i = 1, pos;

if (head == NULL)

printf("\nList is empty\n");

else {

printf("\nEnter index : ");

scanf("%d", &pos);

temp = head;

while (i < pos)

{

ptr=temp;

temp = temp->next;

i++;

}

ptr->next= temp->next;

temp->next->prev=ptr;

free(temp);

}

}