#include<stdio.h>

#include<stdlib.h>

struct node {

int data;

struct node \*next;

};

struct node \*head;

void beginsert();

void lastinsert();

void randominsert();

void begin\_delete();

void last\_delete();

void random\_delete();

void display();

void search();

void main() {

int choice = 0;

while(choice != 9) {

printf("\n\nSingly Linked List Operations\n");

printf("\n1.Insert in begining\n2.Insert at last\n3.Insert at any random location\n4.Delete from Beginning\n");

printf("5.Delete from last\n6.Delete node after specified location\n7.Search for an element\n8.Show\n9.Exit\n");

printf("\nEnter your choice?\n");

scanf("%d", &choice);

switch(choice) {

case 1:

beginsert();

break;

case 2:

lastinsert();

break;

case 3:

randominsert();

break;

case 4:

begin\_delete();

break;

case 5:

last\_delete();

break;

case 6:

random\_delete();

break;

case 7:

search();

break;

case 8:

display();

break;

case 9:

exit(0);

break;

default:

printf("Please enter valid choice..");

}

}

}

void beginsert() {

struct node \*ptr;

int item;

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

if(ptr == NULL) {

printf("\nOVERFLOW");

} else {

printf("\nEnter value\n");

scanf("%d", &item);

ptr->data = item;

ptr->next = head;

head = ptr;

printf("\nNode inserted");

}

}

void lastinsert() {

struct node \*ptr, \*temp;

int item;

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

if(ptr == NULL) {

printf("\nOVERFLOW");

} else {

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

scanf("%d", &item);

ptr->data = item;

if(head == NULL) {

ptr->next = NULL;

head = ptr;

printf("\nNode inserted");

} else {

temp = head;

while (temp->next != NULL) {

temp = temp->next;

}

temp->next = ptr;

ptr->next = NULL;

printf("\nNode inserted");

}

}

}

void randominsert() {

int i, loc, item;

struct node \*ptr, \*temp;

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

if(ptr == NULL) {

printf("\nOVERFLOW");

} else {

printf("\nEnter element value");

scanf("%d", &item);

ptr->data = item;

printf("\nEnter the location after which you want to insert ");

scanf("%d", &loc);

temp = head;

for(i = 0; i < loc; i++) {

temp = temp->next;

if(temp == NULL) {

printf("\ncan't insert\n");

return;

}

}

ptr->next = temp->next;

temp->next = ptr;

printf("\nNode inserted");

}

}

void begin\_delete() {

struct node \*ptr;

if(head == NULL) {

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

} else {

ptr = head;

head = ptr->next;

free(ptr);

printf("\nNode deleted from the beginning ...\n");

}

}

void last\_delete() {

struct node \*ptr, \*ptr1;

if(head == NULL) {

printf("\nlist is empty");

} else if(head->next == NULL) {

head = NULL;

free(head);

printf("\nOnly node of the list deleted ...\n");

} else {

ptr = head;

while(ptr->next != NULL) {

ptr1 = ptr;

ptr = ptr->next;

}

ptr1->next = NULL;

free(ptr);

printf("\nDeleted Node from the last ...\n");

}

}

void random\_delete() {

struct node \*ptr, \*ptr1;

int loc, i;

printf("\n Enter the location of the node after which you want to perform deletion \n");

scanf("%d", &loc);

ptr = head;

for(i = 0; i < loc; i++) {

ptr1 = ptr;

ptr = ptr->next;

if(ptr == NULL) {

printf("\nCan't delete");

return;

}

}

ptr1->next = ptr->next;

free(ptr);

printf("\nDeleted node %d", loc+1);

}

void search() {

struct node \*ptr;

int item, i = 0, flag = 1;

ptr = head;

if(ptr == NULL) {

printf("\nEmpty List\n");

} else {

printf("\nEnter item which you want to search?\n");

scanf("%d", &item);

while(ptr != NULL) {

if(ptr->data == item) {

printf("Item found at location %d", i+1);

flag = 0;

}

i++;

ptr = ptr->next;

}

if(flag) {

printf("Item not found\n");

}

}

}

void display() {

struct node \*ptr;

ptr = head;

if(ptr == NULL) {

printf("Nothing to print");

} else {

printf("\nPrinting values . . . . .\n");

while(ptr != NULL) {

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

ptr = ptr->next;

}

}

}

Singly Linked List Operations

1.Insert in begining

2.Insert at last

3.Insert at any random location

4.Delete from Beginning

5.Delete from last

6.Delete node after specified location

7.Search for an element

8.Show

9.Exit

Enter your choice?

1

Enter value

0

Node inserted

Singly Linked List Operations

1.Insert in begining

2.Insert at last

3.Insert at any random location

4.Delete from Beginning

5.Delete from last

6.Delete node after specified location

7.Search for an element

8.Show

9.Exit

Enter your choice?

2

Enter value?

9

Node inserted

Singly Linked List Operations

1.Insert in begining

2.Insert at last

3.Insert at any random location

4.Delete from Beginning

5.Delete from last

6.Delete node after specified location

7.Search for an element

8.Show

9.Exit

Enter your choice?

3

Enter element value10

Enter the location after which you want to insert 9

Node inserted

Singly Linked List Operations

1.Insert in begining

2.Insert at last

3.Insert at any random location

4.Delete from Beginning

5.Delete from last

6.Delete node after specified location

7.Search for an element

8.Show

9.Exit

Enter your choice?

4

Node deleted from the beginning ...

Singly Linked List Operations

1.Insert in begining

2.Insert at last

3.Insert at any random location

4.Delete from Beginning

5.Delete from last

6.Delete node after specified location

7.Search for an element

8.Show

9.Exit

Enter your choice?

5

Deleted Node from the last ...

Singly Linked List Operations

1.Insert in begining

2.Insert at last

3.Insert at any random location

4.Delete from Beginning

5.Delete from last

6.Delete node after specified location

7.Search for an element

8.Show

9.Exit

Enter your choice?

6

Enter the location of the node after which you want to perform deletion

9

Can't delete

Singly Linked List Operations

1.Insert in begining

2.Insert at last

3.Insert at any random location

4.Delete from Beginning

5.Delete from last

6.Delete node after specified location

7.Search for an element

8.Show

9.Exit

Enter your choice?

7

Enter item which you want to search?

5

Item found at location 5

Singly Linked List Operations

1.Insert in begining

2.Insert at last

3.Insert at any random location

4.Delete from Beginning

5.Delete from last

6.Delete node after specified location

7.Search for an element

8.Show

9.Exit

Enter your choice?

8

Printing values . . . . .

1

2

3

4

5

6

7

8

9

Singly Linked List Operations

1.Insert in begining

2.Insert at last

3.Insert at any random location

4.Delete from Beginning

5.Delete from last

6.Delete node after specified location

7.Search for an element

8.Show

9.Exit

Enter your choice?

9