Operations on Singly Linked List

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

typedef struct linked\_list

{

int data;

struct linked\_list \*next;

} node;

void traversal(node \*start)

{

node \*ptr = start;

printf("The Linked List elements are: ");

while (ptr != NULL)

{

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

ptr = ptr->next;

}

}

void insert\_begin(node \*\*temp)

{

node \*ptr;

int s;

printf("\nEnter the number you want to insert at the beginning: ");

scanf("%d", &s);

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

ptr->data = s;

if (\*temp == NULL)

ptr->next = NULL;

else

ptr->next = \*temp;

\*temp = ptr;

traversal(\*temp);

}

void insert\_end(node \*\*temp)

{

int s;

node \*ptr, \*loc;

printf("\nEnter the number you want to insert at the end: ");

scanf("%d", &s);

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

ptr->data = s;

ptr->next = NULL;

if (\*temp == NULL)

{

\*temp = ptr;

return;

}

loc = \*temp;

while (loc->next != NULL)

loc = loc->next;

loc->next = ptr;

traversal(\*temp);

}

void insert\_mid(node \*\*temp)

{

int s, index, i = 1;

printf("\nEnter the number you want to insert in the middle: ");

scanf("%d", &s);

printf("Enter the index number: ");

scanf("%d", &index);

node \*ptr, \*loc;

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

ptr->data = s;

if (\*temp == NULL)

{

ptr->next = NULL;

\*temp = ptr;

return;

}

loc = \*temp;

while (i < index - 1)

{

loc = loc->next;

if (loc == NULL)

{

printf("Link list size exceed!");

return;

}

i++;

}

ptr->next = loc->next;

loc->next = ptr;

traversal(\*temp);

}

void delete\_first(node \*\*start)

{

node \*ptr;

ptr = \*start;

\*start = ptr->next;

free(ptr);

traversal(\*start);

}

void delete\_last(node \*\*start)

{

node \*ptr, \*q;

ptr = \*start;

while (ptr->next != NULL)

{

q = ptr;

ptr = ptr->next;

}

q->next = NULL;

free(ptr);

traversal(\*start);

}

void deletion\_mid(node \*\*start)

{

node \*loc, \*prev;

int x;

printf("Enter the number you want to delete: ");

scanf("%d", &x);

loc = \*start;

while (loc != NULL)

{

if (loc->data == x)

{

prev->next = loc->next;

free(loc);

printf("\nAfter deleting the node of your choice,\n");

traversal(\*start);

return;

}

prev = loc;

loc = loc->next;

}

printf("Number not present!");

}

void count\_nodes(node \*start)

{

int count = 0;

while (start != NULL)

{

count++;

start = start->next;

}

printf("The total no. of nodes in the linked list are: %d\n", count);

}

void erase\_linkedlist(node \*\*start)

{

node \*old, \*ptr;

ptr = \*start;

while (ptr != NULL)

{

old = ptr;

ptr = ptr->next;

free(old);

}

\*start = NULL;

}

void search(node \*start)

{

int no;

printf("\nEnter the number you want to search: ");

scanf("%d", &no);

node \*ptr = start;

while (ptr != NULL)

{

if (no == ptr->data)

{

printf("The number is present!");

return;

}

else

ptr = ptr->next;

}

}

int main()

{

node \*start, \*ptr, \*temp;

int i, n, s, index, num, opt;

printf("Enter the total no. of elements: ");

scanf("%d", &n);

printf("Enter the first node: ");

scanf("%d", &num);

start = (node \*)malloc(sizeof(node));

start->data = num;

start->next = NULL;

temp = start;

printf("Enter the remaining %d nodes: ", (n - 1));

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

{

scanf("%d", &num);

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

ptr->data = num;

ptr->next = NULL;

temp->next = ptr;

temp = ptr;

}

traversal(start);

printf("\n\nChoose any one from below:\n1) Insert at the beginning\n2) Insertion at the end\n3) Insertion in the middle\n4) Delete the first node\n5) Delete the last node\n6) Delete the node at the given index\n7) Count the no. of nodes\n8) Erase all the nodes\n9) Search an element\n");

printf("Your choice: ");

scanf("%d", &opt);

switch (opt)

{

case 1:

insert\_begin(&start);

break;

case 2:

insert\_end(&start);

break;

case 3:

insert\_mid(&start);

break;

case 4:

delete\_first(&start);

break;

case 5:

delete\_last(&start);

break;

case 6:

deletion\_mid(&start);

break;

case 7:

count\_nodes(start);

break;

case 8:

erase\_linkedlist(&start);

break;

case 9:

search(start);

break;

default:

printf("Invalid choice!\n");

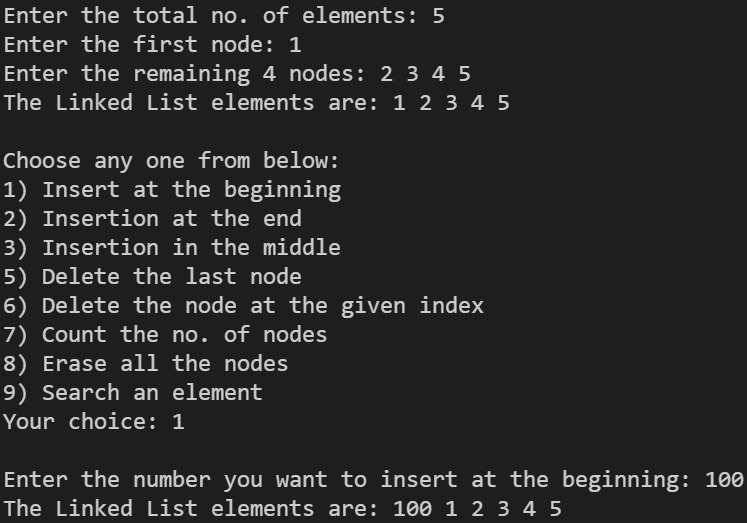
break;

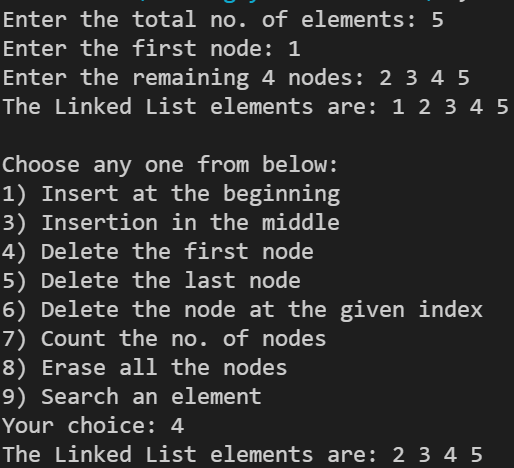
}

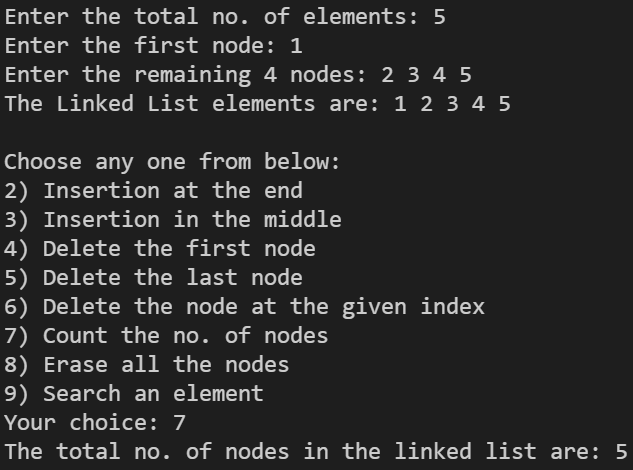
return 0;

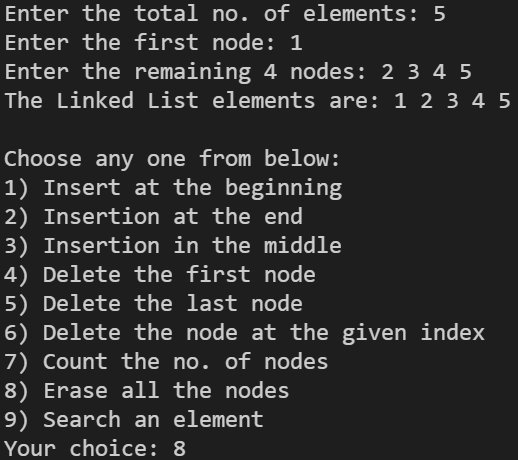
}

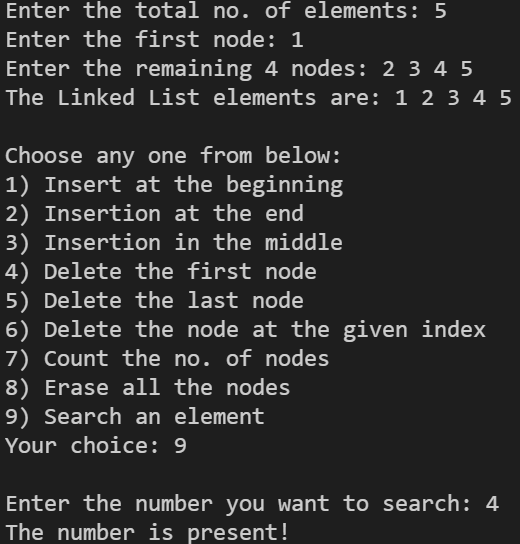
**Outputs:**











Circular Linked List

#include <stdio.h>

#include <stdlib.h>

typedef struct linked\_list{

int data;

struct linked\_list \*next;

} node;

void traversal(node \*start){

node \*ptr;

ptr = start;

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

ptr = ptr->next;

while (ptr != start) {

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

ptr = ptr->next;

}

}

int main(){

int num, n;

node \*start, \*temp, \*ptr;

start = (node \*)malloc(sizeof(node));

printf("Enter the no. of nodes: ");

scanf("%d", &n);

printf("Enter the first node: ");

scanf("%d", &num);

start->data = num;

start->next = start;

printf("Enter the remaining %d nodes: ", (n - 1));

temp = start;

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

scanf("%d", &num);

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

ptr->data = num;

temp->next = ptr;

ptr->next = start;

temp = ptr;

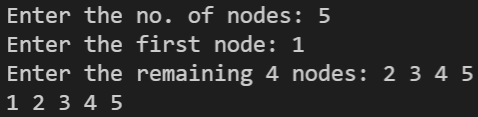
}

traversal(start);

return 0;

}

**Output:**

****