**Write a C and C++ program for implementation of Singly Linked List consisting of**

**following operations:**

**• Inserting a new node in a Linked List at the beginning**

**• Inserting a new node in a Linked List at the end.**

**• Deleting a node from a Linked List from beginning**

**• Deleting a node from a Linked List from end**

**• Display the entire linked list (Traversal)**

**• Now test the working of the program by calling these functions as follows:**

**• Inserting a new node in a Linked List at the beginning: 45**

**• Inserting a new node in a Linked List at the beginning:55**

**• Inserting a new node in a Linked List at the beginning:65**

**• Inserting a new node in a Linked List at the end: 67**

**• Inserting a new node in a Linked List at the end: 77**

**• Inserting a new node in a Linked List at the end: 87**

**• Display the entire linked list (Traversal)**

**• Deleting a node from a Linked List from beginning**

**• Deleting a node from a Linked List from beginning**

**• Display the entire linked list (Traversal)**

**• Deleting a node from a Linked List from end**

**• Deleting a node from a Linked List from end**

**• Display the entire linked list (Traversal)**

**6. Modify the above program (C and C++ both) to make a Menu Driven application. And test the code by invoking the the function as per above given order.**

**7. In the above C code, also add the code of Count the no. of nodes and search a node operation.**

**(these questions are combined under>>)**

#include <stdio.h>

#include <stdlib.h> // Header File for malloc(),calloc(),realloc()

struct node // Structure for the nodes we create

{

int data;

struct node \*next;

};

struct node \*head = NULL; // Head pointer which indicates first node

struct node \*CreateNode(

int value); // function to create a new node, NOTE : this func. won't be

// called directly by the users but by the other functions

void InsertAtBeg(

int value); // function to insert a new node at the beginning of the LL

void InsertAtEnd(

int value); // function to insert a new node at the end of the LL

void InsertAfterNode(

int value,

int after); // function to insert a new node after a given node in the LL

void InsertBeforeNode(

int value,

int before); // function to insert a new node before a given node in the LL

void DeleteFromBeg(); // function to delete a node from the beginning of the LL

void DeleteFromEnd(); // function to delete a node from the end of the LL

void Search(int value); // function to Search an element from the LL

void Display(); // function to display the created LL

int main() {

int choice = 0, value, after, before;

while (choice != 9) {

printf("\n\n\*\*\*\*\*\*\*\*\*\*\*\* MAIN MENU (C) \*\*\*\*\*\*\*\*\*\*\*\*");

printf("\n1 : Add a node at Beginning");

printf("\n2 : Add a node at the End");

printf("\n3 : Add a node after a Node");

printf("\n4 : Add a node before a node");

printf("\n5 : Delete a node from the Beginning");

printf("\n6 : Delete a node from the End");

printf("\n7 : Search an element from the Linked List");

printf("\n8 : Display the Linked List");

printf("\n9 : END the Operation !!!");

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

scanf("%d", &choice);

switch (choice) {

case 1: // calling InsertAtBeg(value) function to add a node in the

// Beginning of the LL

printf("Enter Value to Add : ");

scanf("%d", &value);

InsertAtBeg(value);

printf("\nNode Added at the beginning.");

break;

case 2: // calling InsertAtEnd(value) function to add a node in the End of

// the LL

printf("Enter Value to Add : ");

scanf("%d", &value);

InsertAtEnd(value);

printf("\nNode Added at the end.");

break;

case 3: // calling InsertAfterNode(value,after) function to add a node after

// a node in the LL

printf("Enter Value to Add : ");

scanf("%d", &value);

printf("Enter the Value of node coming before the new node : ");

scanf("%d", &after);

InsertAfterNode(value, after);

printf("\nNode Added after the Node having value %d.", after);

break;

case 4: // calling InsertBeforeNode(value,before) function to add a node

// before a node in the LL

printf("Enter Value to Add : ");

scanf("%d", &value);

printf("Enter the Value of node coming after the new node : ");

scanf("%d", &before);

InsertBeforeNode(value, before);

printf("\nNode Added before the Node having value %d.", before);

break;

case 5: // calling DeleteFromBeg() function to delete a node from the

// Beginning of the LL

DeleteFromBeg();

printf("\nNode Deleted from the Beginning.");

break;

case 6: // calling DeleteFromEnd() function to delete a node from the End of

// the LL

DeleteFromEnd();

printf("\nNode Deleted from the End.");

break;

case 7: // calling Search(value) function to search given element from the

// Linked List

printf("Enter the value you want to search : ");

scanf("%d", &value);

Search(value);

break;

case 8: // calling Display() function to Display the created LL

printf("\nLinked List: ");

Display();

break;

case 9: // exiting from the program

printf("\nExit !!!");

break;

default:

printf("\nInvalid option. Try again.");

break;

}

}

return 0;

}

struct node \*CreateNode(int value) // Creating a new node NOTE : this function

// won't be called directly by the user

{

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

newNode->data = value;

newNode->next = NULL;

return newNode;

}

void InsertAtBeg(int value) // Inserting at beginning

{

struct node \*new\_node = CreateNode(value);

new\_node->next = head;

head = new\_node;

}

void InsertAtEnd(int value) // Inserting at end

{

struct node \*new\_node = CreateNode(value);

if (head == NULL) {

head = new\_node;

} else {

struct node \*ptr = head;

while (ptr->next != NULL) {

ptr = ptr->next;

}

ptr->next = new\_node;

}

}

void InsertAfterNode(int value, int after) // inserting after a node

{

struct node \*new\_node = CreateNode(value);

if (head == NULL) {

head = new\_node;

} else {

struct node \*ptr = head;

while (ptr != NULL && ptr->data != after) {

ptr = ptr->next;

}

new\_node->next = ptr->next;

ptr->next = new\_node;

}

}

void InsertBeforeNode(int value, int before) // inserting before a node

{

struct node \*new\_node = CreateNode(value);

if (head == NULL) {

head = new\_node;

} else {

struct node \*ptr = head;

while (ptr->next != NULL && ptr->next->data != before) {

ptr = ptr->next;

}

new\_node->next = ptr->next;

ptr->next = new\_node;

}

}

void DeleteFromBeg() // Deleting from the Beginning

{

if (head == NULL) {

printf("\nEmpty List !!!");

return;

}

struct node \*ptr = head;

head = head->next;

free(ptr);

}

void DeleteFromEnd() // Deleting from the End

{

if (head == NULL) {

printf("\nEmpty List !!!");

return;

}

struct node \*ptr = head;

while (ptr->next->next != NULL) {

ptr = ptr->next;

}

free(ptr->next);

ptr->next = NULL;

}

void Search(int value) // Searching for an element

{

struct node \*ptr = head;

int pos = 1;

while (ptr != NULL) {

if (ptr->data == value) {

printf("\n%d found at node having position %d\n", ptr->data, pos);

return;

}

ptr = ptr->next;

pos++;

}

printf("\nNode with value %d not found in the LL !\n", value);

}

void Display() // Displaying the list

{

int count = 0; // counting total nodes

struct node \*ptr = head;

if (ptr == NULL) {

printf("List is empty.\n");

return;

}

while (ptr != NULL) {

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

count++; // increasing count by one

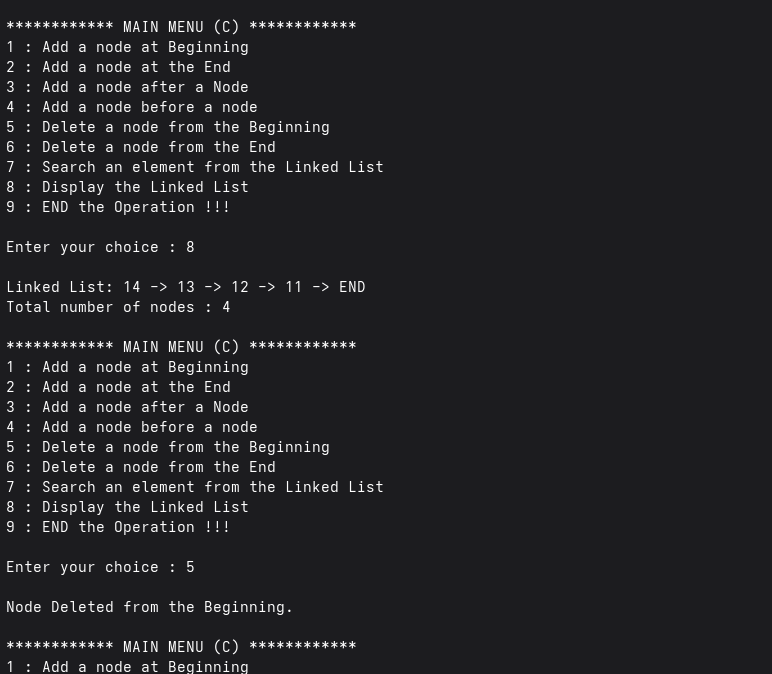
ptr = ptr->next;

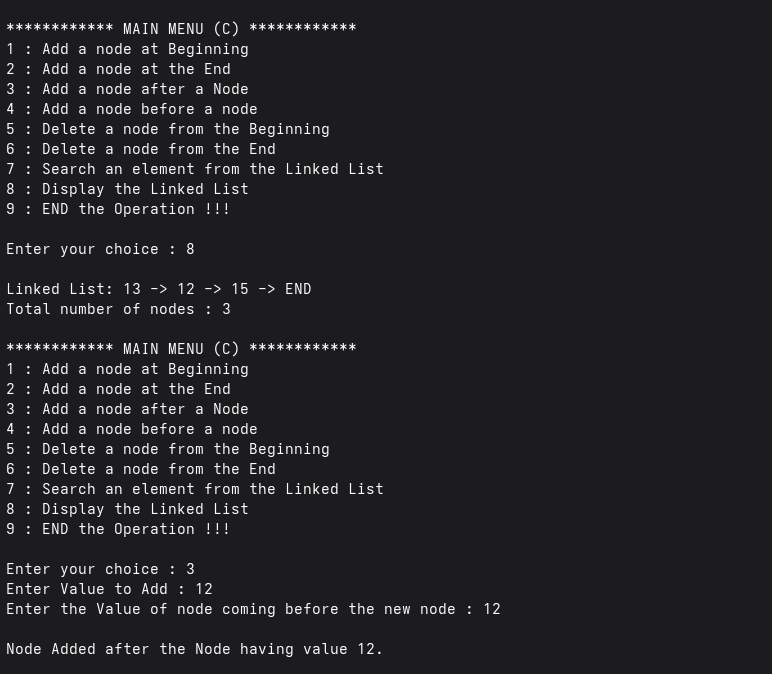
}

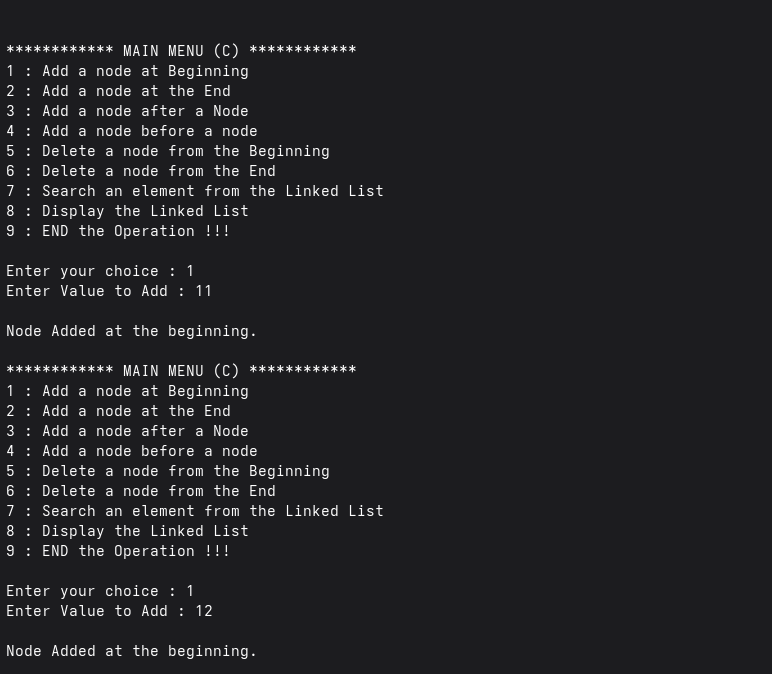
printf("END\n");

printf("Total number of nodes : %d", count);

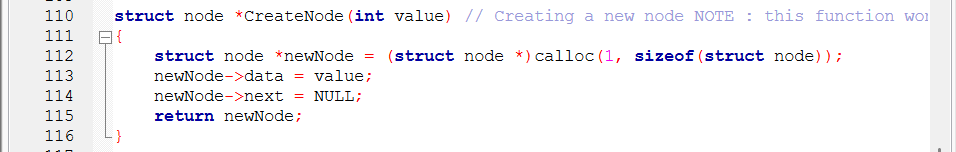
}







**8. Modify the code of C implementation of Question 6 and use calloc() to allocate the memory for a node.**



**NOTE :::::>>> every other code is same as above, thus no ss attached !!**

**9. Write code for a function to create a Node of Linked List and call that function in the insert operations of Linekd List implementation of Question 6.**

**(code for this question is also included in the CreateNode Function used in Question 6.**

**Ss is given above at first image……**

**10. Create a menu-driven C program using a Singly Linked List to manage student records.**

**Each node stores:**

* **Roll No. (treated as PRN)**
* **Name • Phone Number**
* **City**
* **HSC Marks**

**The list supports:**

* **Insert at beginning**
* **Insert at end**
* **Delete from beginning**
* **Delete from end**
* **Display all records**
* **Count records**
* **Search by PRN (Roll No)**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct student {

int roll;

char name[50];

int mobile[10];

char city[25];

float hscmarks;

struct student \*next;

};

struct student \*head = NULL;

struct student \*CreateNode();

void InsertAtBeg();

void InsertAtEnd();

void DeleteFromBeg();

void DeleteFromEnd();

void CountRecords();

void SearchWithROLL();

void DisplayRecords();

int main() {

int choice;

while (choice != 0) {

printf("\n\n\*\*\*\*\*\*\*\*\*\* MENU || STUDENTS RECORDS \*\*\*\*\*\*\*\*\*\*\n");

printf("\n1 : Insert at Beginning");

printf("\n2 : Insert at End");

printf("\n3 : Delete from Beginning");

printf("\n4 : Delete from End");

printf("\n5 : Display Records");

printf("\n6 : Count Total Records");

printf("\n7 : Search by Roll No (PRN)");

printf("\n0 : Exit");

printf("\nEnter Your Choice : ");

scanf("%d", &choice);

switch (choice) {

case 1:

InsertAtBeg();

break;

case 2:

InsertAtEnd();

break;

case 3:

DeleteFromBeg();

break;

case 4:

DeleteFromEnd();

break;

case 5:

DisplayRecords();

break;

case 6:

CountRecords();

break;

case 7:

SearchWithROLL();

break;

case 0:

printf("Exit from menu !\n");

break;

default:

printf("Invalid choice! Try again.\n");

break;

}

}

return 0;

}

struct student \*CreateNode() {

struct student \*NewNode = (struct student \*)malloc(sizeof(struct student));

if (!NewNode) {

printf("Memory allocation Failure!!!\n");

exit(1);

}

printf("Enter Roll No (PRN): ");

scanf("%d", &NewNode->roll);

printf("Enter Name: ");

scanf(" %[^\n]", NewNode->name);

printf("Enter Phone No: ");

scanf(" %[^\n]", NewNode->mobile);

printf("Enter City: ");

scanf(" %[^\n]", NewNode->city);

printf("Enter HSC Marks: ");

scanf("%f", &NewNode->hscmarks);

NewNode->next = NULL;

return NewNode;

}

void InsertAtBeg() {

struct student \*NewNode = CreateNode();

NewNode->next = head;

head = NewNode;

printf("\nRecord inserted at beginning.\n");

}

void InsertAtEnd() {

struct student \*NewNode = CreateNode();

if (head == NULL) {

head = NewNode;

} else {

struct student \*temp = head;

while (temp->next != NULL)

temp = temp->next;

temp->next = NewNode;

}

printf("\nRecord inserted at end.\n");

}

void DeleteFromBeg() {

if (head == NULL) {

printf("List is empty!!\n");

return;

}

struct student \*temp = head;

head = head->next;

printf("\nDeleted record of Roll No: %d\n", temp->roll);

free(temp);

}

void DeleteFromEnd() {

if (head == NULL) {

printf("List is empty!!\n");

return;

}

struct student \*temp = head;

if (head->next == NULL) {

printf("\nDeleted record of Roll No: %d\n", head->roll);

free(head);

head = NULL;

return;

}

while (temp->next->next != NULL) {

temp = temp->next;

}

printf("Deleted record of Roll No: %d\n", temp->next->roll);

free(temp->next);

temp->next = NULL;

}

void CountRecords() {

int count = 0;

struct student \*temp = head;

while (temp != NULL) {

count++;

temp = temp->next;

}

printf("\nTotal number of student records: %d\n", count);

}

void SearchWithROLL() {

if (head == NULL) {

printf("List is empty.\n");

return;

}

int key;

printf("Enter Roll No. to search : ");

scanf("%d", &key);

struct student \*temp = head;

while (temp != NULL) {

if (temp->roll == key) {

printf("Record Found:\n");

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

printf("Name : %s\n", temp->name);

printf("Phone : %s\n", temp->mobile);

printf("City : %s\n", temp->city);

printf("HSC Marks: %.2f\n", temp->hscmarks);

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

return;

}

temp = temp->next;

}

printf("No record found with Roll No: %d\n", key);

}

void DisplayRecords() {

if (head == NULL) {

printf("No records to display.\n");

return;

}

struct student \*temp = head;

printf("\nStudent Records:\n");

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

while (temp != NULL) {

printf("Roll No : %d\n", temp->roll);

printf("Name : %s\n", temp->name);

printf("Phone : %s\n", temp->mobile);

printf("City : %s\n", temp->city);

printf("HSC Marks: %.2f\n", temp->hscmarks);

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

temp = temp->next;

}

}

