SOUTHEAST UNIVERSITY

Meeting the Challenges of Time

Department Of CSE

Cse Assignment

Assignment No:05

Date of submission:

Course name : CSE LAB

Course code : 241

Section : 9

Student's Name :Nazmul Hasan

Student's ID :2023100000130

Submitted To :Mr. Muhammed Yeaseen Morshed Abid

[lecturer, Department of CSE]

Initial : YMA

Code 1:

```
#include<stdio.h>

#include<stdlib.h>


struct node{

   int data;

   struct node *next;

   struct node *prev;

};

int main(){


struct node *head = NULL;

   struct node *m = NULL;

   struct node *n = NULL;

   struct node *o = NULL;

   struct node *p = NULL;

   struct node *q = NULL;


   m = malloc(sizeof(struct node));

   n = malloc(sizeof(struct node));

   o = malloc(sizeof(struct node));

   p = malloc(sizeof(struct node));

   q = malloc(sizeof(struct node));



   printf("Enter data for node m: ");

   scanf("%d", &m->data);
```

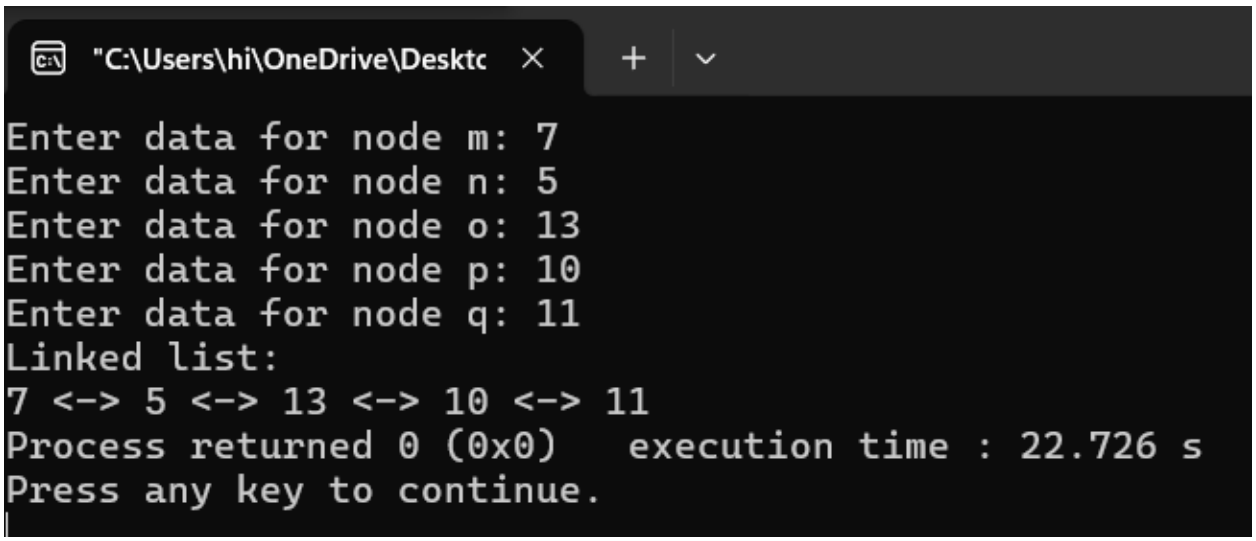
```
printf("Enter data for node n: ");
scanf("%d", &n->data);
printf("Enter data for node o: ");
scanf("%d", &o->data);
printf("Enter data for node p: ");
scanf("%d", &p->data);
printf("Enter data for node q: ");
scanf("%d", &q->data);

head = m;
m->next = n;
m->prev = NULL;
n->next = o;
n->prev = m;
o->next = p;
o->prev = n;
p->next = q;
p->prev = o;
q->next = NULL;
q->prev = p;


printf("Linked list:\n");
struct node *temp = m;
while(temp){
    printf("%d", temp->data);
    if(temp->next){
```

```
        printf(" <-> ");
      }
      temp = temp->next;
    }
    return 0;
}
```

Terminal:



Code 2:

```
#include <stdio.h>
#include <stdlib.h>

// Define the structure for a node in the linked list
struct node {
    int data;
    struct node *next;
    struct node *prev;
```

```
};

// Function to delete the first node that contains a given value
void deleteByValue(struct node **head, int value) {
    struct node *temp = *head;
    struct node *prev = NULL;

    // If the head node itself holds the value to be deleted
    if (temp != NULL && temp->data == value) {
        *head = temp->next; // Change head
        free(temp);         // Free the old head
        return;
    }

    // Traverse the list to find the node containing the given value
    while (temp != NULL && temp->data != value) {
        prev = temp;
        temp = temp->next;
    }
    // If the value was not found in the list
    if (temp == NULL) {
        printf("Value %d not found in the list.\n", value);
        return;
    }

    // Unlink the node from the list
    prev->next = temp->next;
```

```
    // Free memory for the node to be deleted
    free(temp);
}

// Function to print the linked list
void printList(struct node *head) {
    struct node *temp = head;
    printf("Doubly Linked List: ");
    while (temp != NULL) {
        printf("%d", temp->data);
        if (temp->next != NULL) {
            printf(" <-> ");
        }
        temp = temp->next;
    }
    printf("\n");
}
int main() {
    // Initialize nodes
    struct node *head = NULL;
    struct node *m = NULL;
    struct node *n = NULL;
    struct node *o = NULL;
    struct node *p = NULL;
    struct node *q = NULL;
    // Allocate memory for nodes
```

```
m = malloc(sizeof(struct node));
n = malloc(sizeof(struct node));
o = malloc(sizeof(struct node));
p = malloc(sizeof(struct node));
q = malloc(sizeof(struct node));

// Assign data to nodes
m->data = 7;
n->data = 5;
o->data = 13;
p->data = 10;
q->data = 11;

// Connect nodes
head = m;
m->next = n;
m->prev = NULL;
n->next = o;
n->prev = m;
o->next = p;
o->prev = n;
p->next = q;
p->prev = o;
q->next = NULL;
q->prev = p;

// Print the original linked list
```

```
    printf("Original List:\n");
    printList(head);

    // Take user input for the value to delete
    int value;
    printf("Enter the value to delete: ");
    scanf("%d", &value);

    // Delete the node with value
    deleteByValue(&head, value);

    // Print the modified linked list
    printf("\nList after deletion of node with value :\n");
    printList(head);

    return 0;
}
```

Terminal:

```
"C:\Users\hi\OneDrive\Desktc

Original List:
Doubly Linked List: 7 <-> 5 <-> 13 <-> 10 <-> 11
Enter the value to delete: 11

List after deletion of node with value :
Doubly Linked List: 7 <-> 5 <-> 13 <-> 10

Process returned 0 (0x0)   execution time : 15.815 s
Press any key to continue.
```

Code 3:

```
#include <stdio.h>
#include <stdlib.h>

// Define the structure for a node in the linked list
struct node {
   int data;
   struct node *next;
   struct node *prev;
};

// Function to delete a node at a given location (position) in the list
void deleteAtPosition(struct node **head, int position) {
   if (*head == NULL) {
      printf("List is empty.\n");
      return;
   }

   struct node *temp = *head;

   // If the head node itself is to be deleted
   if (position == 0) {
      *head = temp->next; // Change head
      free(temp);        // Free the old head
      return;
   }
```

```
    // Traverse the list to find the node just before the target node
    for (int i = 0; temp != NULL && i < position - 1; i++) {
        temp = temp->next;
    }

    // If position is more than the number of nodes
    if (temp == NULL || temp->next == NULL) {
        printf("Position out of range.\n");
        return;
    }

     // Update pointers to unlink the node from the list
    if (temp->next != NULL) {
        temp->next->prev = temp->prev;
    }
    if (temp->prev != NULL) {
        temp->prev->next = temp->next;
    }

    free(temp); // Free the memory of the node to be deleted
}

// Function to print the linked list
void printList(struct node *head) {
    struct node *temp = head;
    printf("Doubly Linked List: ");
```

```
    while (temp != NULL) {
        printf("%d", temp->data);
        if (temp->next != NULL) {
            printf(" <-> ");
        }
        temp = temp->next;
    }
    printf("\n");
}

int main() {
    // Initialize nodes
    struct node *head = NULL;
    struct node *m = NULL;
    struct node *n = NULL;
    struct node *o = NULL;
    struct node *p = NULL;
    struct node *q = NULL;

    // Allocate memory for nodes
    m = malloc(sizeof(struct node));
    n = malloc(sizeof(struct node));
    o = malloc(sizeof(struct node));
    p = malloc(sizeof(struct node));
    q = malloc(sizeof(struct node));
```

```
// Assign data to nodes
m->data = 7;
n->data = 5;
o->data = 13;
p->data = 10;
q->data = 11;

// Connect nodes
head = m;
m->next = n;
m->prev = NULL;
n->next = o;
n->prev = m;
o->next = p;
o->prev = n;
p->next = q;
p->prev = o;
q->next = NULL;
q->prev = p;

// Print the original linked list
printf("Original List:\n");
printList(head);

// Take user input for the position to delete
int position;
printf("Enter the position to delete: ");
```

```
    scanf("%d", &position);



    // Delete the node at position
    deleteAtPosition(&head, position);

    // Print the modified linked list
    printf("\nList after deletion at position :\n");
    printList(head);



    return 0;
}
```

Terminal:

```
"C:\Users\hi\OneDrive\Desktc

Original List:
Doubly Linked List: 7 <-> 5 <-> 13 <-> 10 <-> 11
Enter the position to delete: 2

List after deletion at position :
Doubly Linked List: 7 <-> 13 <-> 10 <-> 11

Process returned 0 (0x0)   execution time : 2.809 s
Press any key to continue.
```