05-01-2024 Prog-10

Q1 WAP to Implement doubly link list with primitive operations.

a) Create a doubly linked list.

b) Insert a new node to the left of the node.

c) Delete the node based on specific value.

=>

```
#include <stdio.h>
#include <stdlib.h>
Struct Node {
    int data;
    Struct Node * prev;
    Struct Node * next;
};
Struct Node * CreateNode (int data) {
  Struct Node * newNode = (struct Node*) malloc (size of (Struct Node));
  if (newNode == NULL) {
      Printf ("Memory allocation failed \n");
      Exit (1);
   }
  newNode -> data = data;
  newNode -> prev = NULL;
  newNode -> next = NULL;
  return newNode;
}
void insertNode (struct Node ** head, Struct Node* target, int data) {
   Struct Node * newNode = createNode (data);
   newNode -> next = target;
   newNode -> prev = target -> prev;
   if (target -> prev != NULL) {
     target -> prev -> next = newNode;
}
```

05-01-2024

```
    target->prev = newNode;
    if (*head == target) {
        *head = newNode;
    }
}
void deleteNode (struct Node** head, int value) {
    struct Node* current = *head;
    while (current != NULL && current->data != value) {
        current = current->next;
    }
    if (current != NULL) {
        if (current->prev != NULL)
            current->prev->next = current->next;
    }
    if (current->next != NULL) {
        current->next->prev = current->prev;
    }
    if (current->next != NULL) {
        current->next->prev = current->prev;
    }
    if (*head == current) {
        *head = current->next;
    }
    free(current);
} else {
    printf("Node with value %d not found\n", value);
}
}
```

05-02-2024.

```
void printList (struct Node * head) {
    while (head != NULL) {
        printf ("%d <->", head -> data);
        head = head -> next;
    }
    printf ("NULL\n");
}
int main () {
    struct Node* head = CreateNode (1);
    insertNode (&head, head, 2);
    insertNode (&head, head -> next, 3);
    insertNode (&head, head -> next -> next, 4);
    printf ("Doubly linked list:");
    printList (head);
    insertNode (&head, head -> next, 5);
    printf ("After inserting 5 to the left of the second node:");
    printList (head);
    deleteNode (&head, 3);
    printf ("After deleting node with value 3:");
    printList (head);
    return 0;
}
```

O/P

Doubly linked list : 2 <-> 3 <-> 4 <-> 1 <-> NULL

After inserting 5 to the left of the second node :

2 <-> 5 <-> 3 <-> 4 <-> 1 <-> NULL

After deleting node with value 3 :

2 <-> 5 <-> 4 <-> 1 <-> NULL.

```
Doubly Linked List: 2 <-> 3 <-> 4 <-> 1 <-> NULL
After inserting 5 to the left of the second node: 2 <-> 5 <-> 3 <-> 4 <-> 1 <-> NULL
After deleting node with value 3: 2 <-> 5 <-> 4 <-> 1 <-> NULL

Process returned 0 (0x0)   execution time : 0.031 s
Press any key to continue.
```