Week - 06.

01/02/24

// Creating doubly linked list and Inserting a new node to the left of the node and deleting the node based on a specific value.
Displaying the contents of the list.

```
#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
        (sizeof(struct Node));
    if (newNode == NULL)
    {
        printf("Memory allocation failed\n");
        exit(1);
    }
    newNode->data = data;
    newNode->prev = NULL;
    newNode->next = NULL;
    return newNode;
}

void insertNodeToLeft(struct Node* head, struct
Node* target, int data)
{
    struct Node* newNode = createNode(data);
```

```
    if (target -> prev != NULL)
    {
        target -> prev -> next = newNode;
        newNode -> prev = target -> prev;
    }
    else
    {
        head = newNode;
    }
    newNode -> next = target;
    target -> prev = newNode;
}
void deleteNode(struct Node * head, int value)
{
    struct Node* current = head;
    while (current != NULL)
    {
        if (current -> data == value)
        {
            if (current -> prev != NULL)
            {
                current -> prev -> next = current -> next;
            }
            else
            {
                head = current -> next;
            }
            if (current -> next != NULL)
            {
                current -> next -> prev = current -> prev;
            }
            free (current);
            return;
```

```
        }
        current = current->next;
    }
    printf("Node with value %d not found\n",
    value);
}
void displayList(struct Node* head)
{
    printf("Doubly linked list: ");
    while(head != NULL)
    {
        printf("%d <-> ", head->data);
        head = head->next;
    }
    printf("NULL\n");
}
int main()
{
    struct Node* head=NULL;

    head= createNode(18);
    head->next = CreateNode(17);
    head->next->prev=head;
    head->next->next->prev = head->next;

    displayList(head);

    insertNodeToLeft(head, head->next, 7);
    printf("After insertion:\n");
    displayList(head);

    deleteNode(head, 2);
```

```
    printf("After deletion:\n");
    displayList(head);

    return 0;

}
```

Output:

```
Doubly Linked List : 10 <-> 17 <-> 46 <-> NULL
After insertion:
Doubly Linked List : 18 <-> 7 <-> 17 <-> 46 <-> NULL
Node with value 2 not found
After deletion:
Doubly Linked List : 18 <-> 7 <-> 17 <-> 46 <->
NULL
```

// Let code.

```
int getLength(struct ListNode* head){
    int length=0;
    while(head!=NULL){
        length++;
        head=head->next;
    }
    return length;
}
struct ListNode** splitListToParts(struct ListNode*
head, int k, int* returnSize){
    int length=getLength(head);
    int partSize=length/k;
    int remainder=length%k;

    struct ListNode** result=(struct ListNode**)
```

```
malloc(k * sizeof(struct ListNode*));
*returnSize = k;

for(int i=0; i<k; i++){
  int currentPartSize = partSize + (i < re-
  mainder ? 1 : 0);
  if (currentPartSize == 0){
      result[i] = NULL;
  }
  else{
    result[i] = head;
    for(int j=0; j< currentPartSize-1; j++)
    {
      head = head->next;
    }
    struct ListNode* temp = head->next;
    head->next = NULL;
    head = temp;
  }
}
return result;
}
```

output :-

case ①

head = [1, 2, 3]

k = 5

=> [[1], [2], [3], [], []]

case ②:

head = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

k = 3

Output :-

[[1,2,3,4], [5,6,7], [8,9,10]]