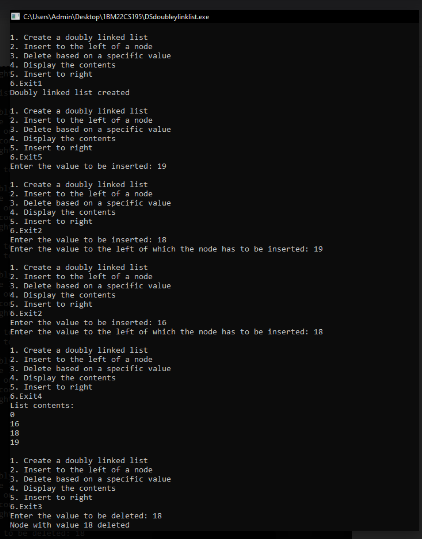
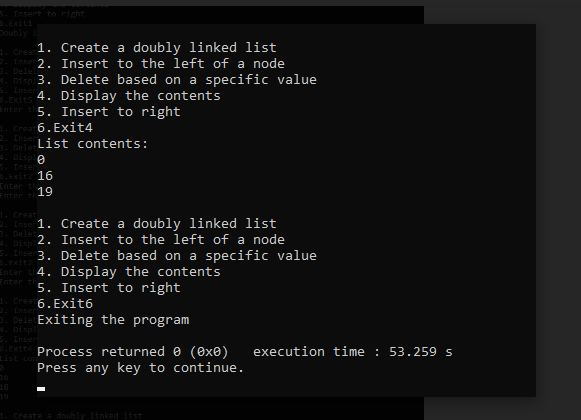
Lab 7 Doubly link list

#include<stdio.h>  
#include<stdlib.h>  
struct node {  
    int data;  
    struct node \*prev;  
    struct node \*next;  
};  
struct node \*s1 = NULL;  
struct node \*createNode(int value) {  
    struct node \*temp = (struct node \*)malloc(sizeof(struct node));  
    temp->data = value;  
    temp->next = NULL;  
    temp->prev = NULL;  
    return temp;  
}  
struct node \*insert\_left(struct node \*start) {  
    int value, key;  
    struct node \*temp = createNode(0);  
    printf("Enter the value to be inserted: ");  
    scanf("%d", &temp->data);  
    printf("Enter the value to the left of which the node has to be inserted: ");  
    scanf("%d", &key);  
    struct node \*ptr = start;  
    while (ptr != NULL && ptr->data != key) {  
        ptr = ptr->next;  
    }  
    if (ptr == NULL) {  
        printf("Node with value %d not found\n", key);  
        free(temp);  
    } else {  
        temp->next = ptr;  
        temp->prev = ptr->prev;  
        if (ptr->prev != NULL) {  
            ptr->prev->next = temp;  
        }  
        ptr->prev = temp;  
        if (ptr == start) {  
            start = temp;  
        }  
    }  
    return start;  
}  
struct node \*delete\_value(struct node \*start) {  
    int value;  
    printf("Enter the value to be deleted: ");  
    scanf("%d", &value);  
    struct node \*ptr = start;  
    while (ptr != NULL && ptr->data != value) {  
        ptr = ptr->next;  
    }  
    if (ptr == NULL) {  
        printf("Node with value %d not found\n", value);  
    } else {  
        if (ptr->prev != NULL) {  
            ptr->prev->next = ptr->next;  
        } else {  
            start = ptr->next;  
        }  
        if (ptr->next != NULL) {  
            ptr->next->prev = ptr->prev;  
        }  
        printf("Node with value %d deleted\n", value);  
        free(ptr);  
    }  
    return start;  
}  
void display(struct node \*start) {  
    struct node \*ptr = start;  
    if (start == NULL) {  
        printf("List is empty\n");  
    } else {  
        printf("List contents:\n");  
        while (ptr != NULL) {  
            printf("%d\n", ptr->data);  
            ptr = ptr->next;  
        }  
    }  
}  
  
struct node \*insert\_right(struct node \*start) {  
    int value;  
    struct node \*temp = createNode(0);  
    printf("Enter the value to be inserted: ");  
    scanf("%d", &temp->data);  
  
    if (start == NULL) {  
        start = temp;  
    } else {  
        struct node \*ptr = start;  
        while (ptr->next != NULL) {  
            ptr = ptr->next;  
        }  
        ptr->next = temp;  
        temp->prev = ptr;  
    }  
  
    return start;  
}  
  
int main() {  
    int choice;  
    while (1) {  
        printf("\n1. Create a doubly linked list\n2. Insert to the left of a node\n3. Delete based on a specific value\n4. Display the contents\n5. Insert to right\n6.Exit");  
        scanf("%d", &choice);  
        switch (choice) {  
            case 1:  
                s1 = createNode(0);  
                printf("Doubly linked list created\n");  
                break;  
            case 2:  
                s1 = insert\_left(s1);  
                break;  
            case 3:  
                s1 = delete\_value(s1);  
                break;  
            case 4:  
                display(s1);  
                break;  
            case 5:  
                s1 = insert\_right(s1);  
                break;  
            case 6:  
                printf("Exiting the program\n");  
                exit(0);  
            default:  
                printf("Invalid choice\n");  
        }  
    }  
    return 0;  
}

Output:-





Leet code :

Singly link list

#include <stdlib.h>

struct ListNode\*\* splitListToParts(struct ListNode\* head, int k, int\* returnSize) {

struct ListNode\* current = head;

int length = 0;

while (current) {

length++;

current = current->next;

}

int part\_size = length / k;

int extra\_nodes = length % k;

struct ListNode\*\* result = (struct ListNode\*\*)malloc(k \* sizeof(struct ListNode\*));

current = head;

for (int i = 0; i < k; i++) {

struct ListNode\* part\_head = current;

int part\_length = part\_size + (i < extra\_nodes ? 1 : 0);

for (int j = 0; j < part\_length - 1 && current; j++) {

current = current->next;

}

if (current) {

struct ListNode\* next\_node = current->next;

current->next = NULL;

result[i] = part\_head;

current = next\_node;

} else {

result[i] = NULL;

}

}

\*returnSize = k;

return result;

}

Output:-

