Date 1/01/24.
Page

WEEK 7:-

Doubly linked list

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

typedef struct node {
    int data;
    struct node * next;
    struct node * prev;
} node;

node * head = NULL;
int count = 0;

void insert (int data, int position);
void delete (int element);
void display ();

int main () {
    int data, choice, pos;
    printf ("1. Insert \n 2. Delete \n 3. Exit \n Choice:");
    scanf ("%d", &choice);
    while (choice != 3) {
        if (choice == 1) {
            printf ("Enter data and position: ");
            scanf ("%d %d", &data, &pos);
            insert (data, pos);
            printf ("Count : %d \n", count);
        } else if (choice == 2) {
            printf ("Enter element :");
```

```
        scanf("%d", &pos);
        delete(pos);
        printf("count: %d\n", count);
    }
    display();
    printf("Enter choice:");
    scanf("%d", &choice);
}
return 0;
}
void insert(int data, int position){
    if (position == 0){
        node* new_node = malloc(sizeof(node));
        new_node -> data = data;
        new_node -> next = head;
        new_node -> prev = NULL;
        if (head != NULL) head -> prev = new_node;
        head = new_node;
        count++;
        return;
    } else if (position == count){
        node* new_node = malloc(sizeof(node));
        new_node -> data = data;
        new_node -> next = NULL;
        node* temp = head;
        while (temp -> next != NULL)
            temp = temp -> next;
        temp -> next = new_node;
        new_node -> prev = temp;
        count++;
        return;
    } else if (position > count || position < 0){
```

```
printf ("Unable to insert at given position \n");
return;
} else {
    node *temp = head;
    for (int i = 0; i < position - 1; i++)
      temp = temp -> next;
    node* new node = malloc (sizeof (node));
    new_node -> data = data;
    new_node -> next = temp -> next;
    new_node -> prev = new_node;
   temp -> next = new_node;
    count ++;
   return;
  }
}.
void delete (int element) {
    int position = 0; node * temp = head;
    if (head == NULL) {
      printf (" Element does not exist in list ");
      return;
    }.
  if (position == 0) {
    node * temp = head;
    temp = temp -> next;
    temp -> prev = NULL;
     free (head);
     head = temp;
     count --;
    return;
  } else if (position == count -1) {
    node* temp = head;
    for (int i = 1; i < count -1 ; i ++).
```

```
        temp = temp -> next;
        temp -> prev = NULL;
        free(head);
        head = temp;
        count--;
        return;
    } else if (position == count - 1) {
        node* temp = head;
        for (int i = 1; i < count - 1; i++)
            temp = temp -> next;
        node* temp1 = temp -> next;
        temp -> next = NULL;
        free(temp1);
        count--;
        return;
    } else if (position >= count || position < 0) {
        printf("Unable to delete at position\n");
        return;
    } else {
        node* temp = head;
        for (int i = 0; i < position; i++)
            temp = temp -> next;
        temp -> next -> prev = temp -> prev;
        temp -> prev -> next = temp -> next;
        free(temp);
        count--;
        return;
    }
}
void display() {
    node* temp = head;
    printf("Linked list :\n");
```

Date __/__/__
Page _____

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

OUTPUT:-

1. Insert
2. Delete
3. Exit

Choice : 1
Enter data & position : 2 0
Count : 1
Enter choice : 1
Enter data & position : 3 @ 1
Count : 2.
Linked list : 2 3
Enter choice : 1
Enter data & position : 4 0
Count : 3
Linked list : 4 2 3
Enter choice : 1
Enter data and position : 5 1
Count : 4.
Linked list : 4 5 2 3
Enter choice : 2
Enter element : 2
Count : 3
Linked list : 4 5 3
Enter choice : 3
Exit

Date / /
Page

OUTPUT:- LEETCODE :-

```
struct ListNode** splitListToParts(struct ListNode* head,
int k, int* returnSize) {
    struct ListNode* temp = head, int n = 0;
for(; temp != NULL; temp = temp->next, n++);
struct ListNode** lists = (struct ListNode**) malloc.
                          (k * sizeof(struct ListNode*));
for(int i = 0; i < k; i++) list[i] = NULL;
int earlier_lists = n % k; size = n / k;
int current = 0; bool last_over = false;
temp = head;
*returnSize = k;
for(int i = earlier_lists; i > 0; i--) {
        struct ListNode* temp1 = temp;
        lists[current++] = temp;
        for(int j = 0; j < size; j++) temp1 = temp1->next;
        temp = temp1->next;
        temp->next = NULL;
}.
if(temp == NULL) return lists;
for(int i = 0; i < k - earlier_list; i++) {
        struct ListNode* temp1 = temp;
        if(temp1 == NULL) break;
        for(int j = 0; j < size - 1; j++)
        temp1 = temp1->next;
        temp = temp1->next;
        temp1->next = NULL;
}
return lists;
}.
```