Date 10/01/24
Page

5. ~~#include~~ WAP to implement singly linked list with following operation
a) create a Linked list
b) deletion of node at begging, at specified ~~position~~ element & at the end
c) display the contents of linkedlist

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

struct node{
    int data;
    struct Node*next;
    };

struct Node* createNode(int data){
    struct Node* newNode = (struct Node*) malloc(
    sizeof(struct Node));
    if(!newNode){
        printf("memory allocation failed\n");
        exit(1);
    }
    newNode -> data = data;
    newNode -> next = NULL;
    return newNode;
}

void append(struct Node** head_ref, int data){
    struct Node* newNode = createNode(data);
    if(*head_ref == NULL){
        *head_ref = newNode;
    } else{
```

```
struct Node* temp = *head_ref;
while (temp->next != NULL){
  temp = temp->next;
}
temp -> next = newNode;
}
printf("Added %d to the linked list\n",
      data);
}
void deleteFirst(struct Node** head_ref
{
  if(*head_ref == NULL) {
    printf("List is empty. No element to
  delete.\n");
    return;
  }
  struct Node* temp = *head_ref;
  *head_ref = temp->next;
  printf("Deleted first element: %d\n",
  temp->data);
  free(temp);
}

void deleteLast(struct Node** head_ref) {
  if(*head_ref == NULL) {
    printf("Deleted Last element. List i
  empty. No element to delete.\n");
    return;
  }
  if((*head_ref)->next == NULL){
    printf("Deleted Last element: %d\n",
    *head_ref)->data);
```

```
free(*head_ref);
*head_ref = NULL;
return;
}
struct Node* temp = *head_ref;
while (temp -> next -> next != NULL){
    temp = temp -> next;
}
printf ("Deleted last element : %d\n", temp->
next -> data);
free(temp->next);
temp -> next = NULL;
}

void deleteElement (struct Node** head_ref, int
    {
    if (*head_ref = NULL){
    printf("List is empty. No element to
        delete.\n");
    return;
    }
    struct Node* temp = *head_ref;
    if(temp != NULL && temp -> data == key){
        *head_ref = temp->next;
        printf (" Deleted specified element : %d\n",
            key);
        free(temp);
        return;
    }
    struct Node* prev = NULL;
    while (temp != NULL && temp -> data != key)
    { prev = temp;
        temp = temp->next;
    }
```

```
if(temp == NULL) {
printf ("Element %d not found in the
list \n", key);
return;
}
prev->next = temp->next;
printf ("Deleted Specified element %d\n",
free(temp);
}

void display(struct Node* head) {
if (head == NULL){
    printf("List is empty\n");
    return;
}
struct Node*temp = head;
while (temp != NULL) {
    printf ("%d ->", temp->data);
    temp = temp->next;
}
printf("NULL \n");
}

int main()
{ struct node*head = NULL;
int choice, value;
Printf("
while(1) {
printf("() menu:");
Printf ("Enter 1 to add the element:")
printf("Enter 2 to delete at First:")
Printf("Enta 3 to delet, at Last:)
```

```
printf("Enter 4 to delete the element at position:");
printf("Enter 5 to display the Linked list");
printf("Exit:");
printf
printf("Enter your option:");
scanf("%d", &choice);
switch (choice)
{
 case 1:
    printf("Enter value:");
    scanf("%d", &value);
    append(&head, value);

 case 2:
    deleteFirst(&head);
 case 3:
    deleteLast(&head);
 case 4:
    printf("Enter position:");
    scanf("%d", &value);
    deleteElement(&head, value);
    case 5:
        display();
 }
}
return 0;
}
```

Dynamic memory allocation
malloc
calloc

1. Insertion at beggining
2. Insertion at end
3. Deletion at beginning
4. Deletion at end
5. Deletion at Specified Position
6. Display list

Enter your choice: 1
Enter data to insert at beggining: 1

Executed
11/11

Enter your choice: 2
Enter data to insert at end: 22

Enter your choice: 6
11 12

Enter your choice: 3
Data is deleted at beginning

Enter your choice: 4
Data is deleted at ending

Enter your choice: 6
List is empty

Enter your choice: 1
Enter data to insert at beggin
2 2

Enter your choice: 5
Enter position to delete 0

Enter your choice 6
List is empty