22/1/24

1) Wap to implement Singly linked list with following operation:

a) Create a linked list
b) Insertion of node at first position, at any position and at end of list.
c) Display the contents of the linked list.

→

```
#include <stdio.h>
#include <math.h>
#include <string.h>

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

struct node * createNode (int data) {

    struct node * newNode;
    newNode = (struct node*) malloc (sizeof (struct node));
    if (newNode == NULL) {
        printf ("Memory allocation failed");
        exit (1);
    }
    newNode -> data = data;
    newNode -> next = NULL;
    return newNode;
}
```

```
struct node * cr-ll ( int value[], int size) {

    struct node *head = NULL;
    struct node * tail = NULL;

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

    Struct node* newNode = createNode
                           (values[i]);

  if (head == NULL) {
      head = newNode;
      tail = newNode;
    }
  else {
      tail -> next = newNode;
      tail = newNode;
    }
  }
  return head;
}

void insertFirst ( struct node **head, int data) {

    struct node* newNode;
    newNode = createNode (data);
    newNode -> next = *head;
    *head = newNode;
}
```

```
void insertAtPos(struct node **head, int data,
                 int position){

    if(position==0){

        insertFirst(head, data);
        return;
    }

    struct node *new_node = createNode(data);
    struct node *current = *head;

    for(int i=0; i<position-1; ++i){

        if(current==NULL){
            printf("Invalid position\n");
            return;
        }

        current = current->next;
    }

    new_node->next = current->next;
    current->next = new_node;
}

void insertEnd(struct node **head, int data){

    struct node *new_node = createNode(data);

    if(*head==NULL){
        *head = new_node;
        return;
    }
```

```
    struct node * current = *head;
    while ( current -> next != NULL) {
        current = current -> next;
    }
    current -> next = new_node;
}

void display ( struct node *head ) {

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

int main () {

    int values = { 1, 2, 3, 4 };
int size = sizeof (                    );
    struct node * linked list = createLinkedList();

    int data;
    printf ( " Enter data to insert at the beginning: " );
    scanf ( " %d ", & data);
    insertFirst ( & linkedList, data);

    int position;
    printf ( " Enter data to insert at specific position: " );
    scanf ( " %d ", & data);
```

```
printf("Enter the position: ");
scanf("%d", &position);
insertAtpos(&linkedlist, data, position);

printf("Enter data to insert at end: ");
scanf("%d", &data);
insertEnd(&linkedList, data);

display(linkedList);

return 0;
}
```

Output:

```
Enter the number of elements : 6
Enter the elements:
1
2
3
4
5
6
Enter data to insert at the beginning : 58
Enter data to insert at specific position: 69
Enter the position : 5
Enter data to insert at the end : 100
```

58 → 1

58, 1, 2, 3, 4, 69, 5, 6, 100, NULL

2) WAP to implement Singly Linked list with following operation:

a) Create a linked list
b) Deletion of first element, specific element and last element in the list.
c) Display the contents of the linked list.

→

```
#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 == NULL) {
        printf("Memory allocation failed\n");
        exit(1);
    }

    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}
```

```
struct node * create() {

    struct node * head = NULL;
    struct node * tail = NULL;
    int size, data;

printf(" Enter the no. of elements : ");
scanf("%d", &size);

printf(" Enter the elements \n");
  for(int i=0 ; i < size ; ++i){
    scanf("%d", &data);

struct node * newNode = createNode(data);

    if ( head == NULL){

        head = newNode;
        tail = newNode;
      }
      else {
        tail->next = newNode;
        tail = newNode;
    }
}

    return head;
}
```

```
void deleteFirst ( struct node **head ) {

    if ( *head == NULL) {
        printf ("List is empty. Nothing to
                delete ");
        return ;
    }

    struct node *temp = *head;
    *head = (*head) -> next ;
    free ( temp);
}

void deleteElement ( struct Node **head , int key) {

    if ( *head == NULL ) {
    printf ("List is Empty . Nothing to delete. ");
        return ;
    }
    struct node * current = *head;
    struct node * prev = NULL;

    while ( current != NULL && current -> data !=
                key) {
        prev = current;
        current = current -> next ;
    }
    if ( current == NULL ) {
        printf (" Element not found in the list ");
        return ;
    }
```

```
    if (prev == NULL) {
        *head = current -> next;
    }
    else {
        prev -> next = current -> next;
    }
    free(current);
}

void deleteLast (struct node **head) {
    if (*head == NULL) {
        printf("List is empty. Nothing to delete.");
    }
    if ((*head) -> next == NULL) {
        free(*head);
        *head = NULL;
        return;
    }

    struct node *current = *head;
    struct node *prev = NULL;

    while (current -> next != NULL) {
        prev = current;
        current = current -> next;
    }
    prev -> next = NULL;
    free(current);
}
```

```
void display ( struct node * head) {

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

int main (){

    struct node *ll = cr-ll();

    printf (" Linked list before deletion: ");
     display ( ll);

    deleteFirst (&ll);
    printf (" Linked list after deleting first element \n");
     display ( ll);

    int key;
printf (" Enter the element to delete ");
scanf ("%d", & key);
 deleteElement ( & ll, key);
printf (" Linked list after deleting specific element \n");
display (ll);

 deletelast ( & ll);
printf (" Linked list after deleting last element \n");
 display ( ll);
return 0;
}
```

Output

Enter the number of elements : 6;
Enter the elements :
1
3
4
5
6
2
Linked list before deletion :
1, 3, 4, 5, 6, 2, NULL

Linked list after deleting first element:
3, 4, 5, 6, 2, NULL

Enter the element to delete : 4.

Linked list after deleting specific element :
3, 5, 6, 2, NULL

Linked list after deleting last element :
3, 5, 6, NULL.

22/11/24