2/12/25 a) WAP to implement doubly linked list with primitive operations.
(1) Create a doubly link list.
(2) Insert at left and right.
(3) Delete node based on value and position
(4) Display the contents.

```
(1) Struct Node* createNode (int data){
        Struct Node* newNode = (Struct Node*)malloc (Sizeof(Struct Node));
        newNode -> data = data;
        newNode -> prev = NULL;
        newNode -> next = NULL;
        return newNode; }

(2) Void insertStart (int data){
        Struct Node* newNode = createNode(data);
        if ( head==NULL){
            head = newNode;
            return ; }
        newNode -> next = head;
        head -> prev = newNode;
        head = newNode; }
    Void insertEnd (int data){
        Struct Node* newNode = createNode(data);
        if (head==NULL){
            head = newNode;
            return;}
        Struct Node* temp = head;
        while (temp->next!=NULL) {
            temp = temp->next; }
```

```
temp→ next = newNode;
newNode → prev = temp; }
void Insertafterof Position ( int data, int pos){
if (pos <=1){
    insert Start(data); }
Struct Node* temp = head;
 int count = 1;
while (temp!= NULL && count < pos-1){
   temp = temp → next;
   Count ++; }
if (temp == NULL){
  insert End(data); }
        newNode → next = temp;
        newNode → prev = temp → prev;
        if( temp → prev != NULL){
           temp → prev → next = newNode; }
        temp → prev = newNode;

(3)  Void deletebyValue( int value){
  Struct Node* ~~newNode~~ temp =head;
  while(temp!=NULL && temp→ data!= value){
     temp = temp → next; }
  if (temp == NULL){
     printf (" Value not found ");
     ~~return~~; }
  if (temp → prev!=NULL){
     temp → prev → next = temp → next; }
 else {  head = temp → next; }
  if ( temp → next! = NULL){
     temp → next → prev = temp → prev; }
free(temp); }
```

2

```
void deleteposition (int pos){
  if (head == NULL){
      printf (" List is empty ");
      return; }
   Struct Node *temp = head;
   int count =1;
   if (pos = =1){
     head =head→ next;
     if (head != NULL)
         head → prev = NULL;
         free (temp);
         return; }
  While ( temp != NULL && count < pos){
       temp = temp → next;
       Count ++ ; }
 if (temp==NULL){
     printf (" Out of range");
      return; }
 if ( temp → prev != NULL ){
     temp→prev → next = temp → next; }
 if ( temp→next!= NULL ){
     temp→next→prev = temp → prev; }
 free (temp); }
```

Execute

O/P 1. create doubly linked list
2. Insert at start
3. Insert at end
4. Insert at ~~position~~ left of position
5. Delete by value
6. Delete by position
7. Display
8. Exit .

```
→ Enter your choice : 1
  How many values ? 4
  Enter value 1: 10
  Enter value 2: 20
  Enter value 3: 30
  Enter value 4: 40

→ Enter your choice: 2
  Enter value : 15

→ Enter your choice : 3
  Enter value : 25

→ Enter your choice : 4
  Enter value : 35
  Enter position : 4

→ Enter your choice : 7
Doubly linked list:
  15↔10↔35↔20↔30↔40↔25↔NULL

→ Enter your choice : 5
  Enter value : 40

→ Enter your choice : 6
  Enter position to delete : 2

→ Enter your choice : 7
  doubly linked list :
  15↔35↔20↔30↔25↔ NULL

→ Enter your choice : 8
  Exiting.
```

[illegible]