1.) WAP to impliment doubly link list with primitive operations.

a) create double linked list

b) Insert a new node to left of node

c) Delete the node based on specific value.

```
#include < stdio.h>
#include<stdlib.h>
struct node {
        struct node* next;
        int data;
        struct node *prev;
};
struct node * start =NULL;
struct node * create_ll (struct node * );
struct node * insert_before (struct node*);
struct node * del_specific (struct node*);

struct node * create_ll (struct node * start ){
        struct node * newnode, * ptr ;
        int num;
        Printf (" Enter -1 to end ");
        printf (" Enter a value : ");
        Scanf ( "d ", & num);
        while (num != -1 ) {
                if (start = = null )
                {
                newnode = (struct node*) malloc (sizeof (struct node));
                newnode -> prev = NULL;
newnode → data = num;
newnode → next = null;
Start = newnode;
}
else {
      ptr = start;
      newnode = (struct node*)malloc( size of (struct node));
      newnode → data = num;
      while (ptr → next != NULL)
         {
            ptr = ptr → next;
         }
        ptr → next = newnode;
        newnode → prev = ptr;
        newnode → next = NULL;
    }
   printf("\n enter the data: ");
   scanf ("%d",&num);
 }
 return start;
};
Struct node * insert_before (struct node * start) {
      struct node * ptr, * newnode;
      int num, val;
      printf(" Enter the data: ");
      scanf ("%d", &num);
      printf(" Enter the value before which data is inserted: ");
```

```
newnode → data = num;
ptr = start;
while ( ptr → data != val ) {
      ptr = ptr → next;
  }
newnode → next = ptr;
newnode → prev = ptr → prev;
ptr → prev → next = newnode;
ptr → prev = newnode;
return start;
};
struct node * del_specific (struct node* start ){
        int num, ~~counter~~; value;
        printf ("Enter ~~position~~ to be deleted: ");
        scanf ( "%d", &~~num~~ value );
        if ( start == NULL) {
              printf ("LL is empty \n");
              return start; }
        else {  struct node * ptr;
              ptr = start;
              while (ptr → data != value ) {
                     ptr = ptr → next; }
              ptr → prev → next = ptr → next;
              ptr → next → prev = ptr → prev;
              free (ptr);
              Print ("value deleted");
              return start;
            }
        };
```

```
void main(){
    it choice ;
    Print F("1: create LL \n 2: Insert node on left \n
          3: ~~insert nor~~ delete node at specific position );
   while (1) {
      printF("Enter your choise:");
      scanF("%d", & choise);
      stitch (choice )
         {
         case 1: start = create_ll (start);
                 break;
        case 2: start = insert_left (start);
                 break;
        case 3: start = del_specific (start);
                 break;
       default: exit (0);
      }
   }
}
```

Exect