5/2/2024

1) WAP to impliment doubly linked list with primitive operation:

ⓐ Create a doubly linked list

→
```
struct node {
    int data;
    struct node *prev;
    struct node *next;
};
struct node *head, *tail;
head = 0;

void create_dll(){

    struct node *newnode;
newnode = (struct node *) malloc (sizeof (struct node));
    printf("Enter data: ");
    scanf("%d", &newnode -> data);
    newnode -> next = 0 ;
    newnode -> prev = 0 ;

    if ( head == 0) {
        head = tail = newnode;
    }
    else {  newnode -> prev = tail ;
            tail -> next = newnode;
            tail = newnode;
    }
}
```

(b) Insert a newnode to the left of a node.

→ void in_at_left () {

```
        Struct node *newnode ~~{~~, *ptr; ptr=head;
newnode = (struct node *) malloc (sizeof (struct node));
        printf ("Enter data: ");
        scanf ("%d", &newnode -> data);
      int count=0 , pos;
        printf (" Enter position: ");
        scanf(" %d ", & pos);

      while ( count < pos) {
            ptr = ptr -> next ;
            count ++ ;
         }

        ~~ptr->prev -> next = newnode;~~
          ~~ptr~~ newnode -> prev = ptr->prev ;
        newnode -> next = ptr;
        ptr -> prev -> next = new node ;
              ptr -> prev = new node;
      }
```

ⓒ Delete the node based on a specific value.

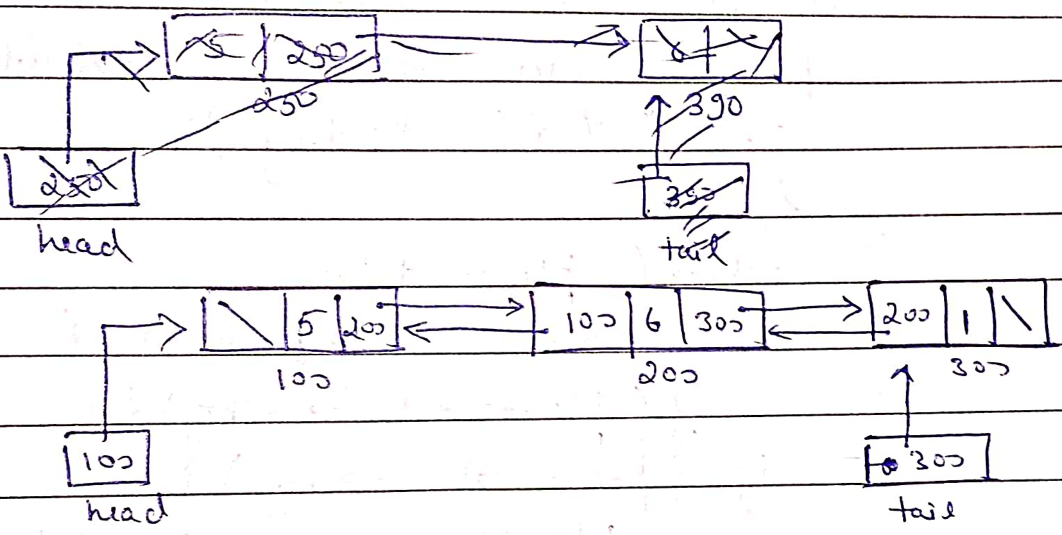
```
→    void del-at-pos(){

          struct node *ptr;  ptr = head;
     int key, val, flag = 0;
      printf("Enter value to be deleted: ");
      scanf("%d", &key);
     ~~if(~~ ~~[illegible]~~
        ~~while( ptr -> data~~
       while ( ptr ~~-> [illegible]~~ != NULL){
           ~~if( ptr -> data != key) {~~
          if (ptr -> data == key) {
                ptr -> prev -> next = ~~[illegible]~~ ptr -> next;
               ptr -> next -> prev = ptr -> prev;
               ~~free(ptr);~~   flag = 1;
             ~~}~~    ~~break;~~ ~~flag = 1;~~  free(ptr);
     ~~ptr = else {~~   ~~ptr -> next;~~   break; }
          else {
                   ptr = ptr -> next;
                }
          }
       if (flag == 1){
               printf("Element deleted");
          }
       else {
               printf(" Element not found");
          }
     }
```

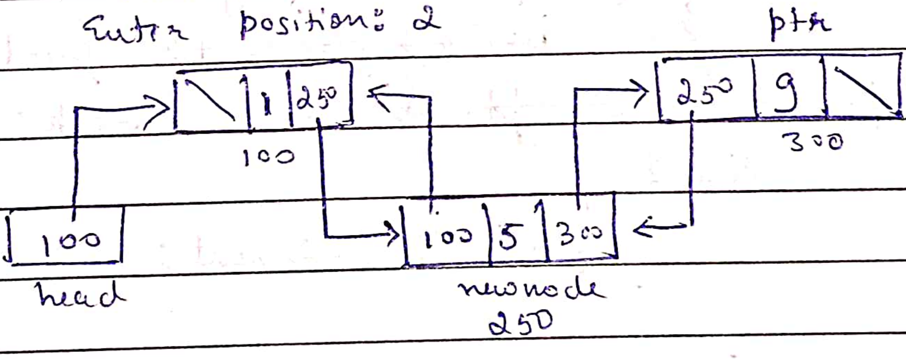
Output

a) Enter data: 5
Enter data: 6
Enter data: 1

newnode will be created like this.



b) Enter data: 5
Enter position: 2



c) Enter value to be deleted: 5

Before: 2 3 5 1
After: 2 3 1

~~Before~~

Date
Page

for

(b)

```
Enter data : 5
Enter position: 2

Before :  1    9
After  :  1    5    9
```

[illegible]
5/2/24

(leet code)

(5) Leet code program 1. (Score of Parenthesis)

→
```
int scoreOfParentheses (char *s) {

    int score = 0 ;
    int depth = 0 ;

  for (int i=0 ; s[i] != '\0' ; i++) {
        if (s[i] == '(' ) {
            depth ++ ;
      }
      else {
          depth -- ;
          if (s[i-1] == '(') {
              score += 1 << depth
          }
        }
      }
    return score ;
  }
```