

```

Class List
{
    private:
        node * head;
        int size;
    public:
        List()
        {
            head = NULL;
            size = 0;
        }
        void insert(int val, int pos);
};

```

Insertion at head :  $\text{insert}(2, \text{pos} \rightarrow \underline{1})$  1 → onwards

size = 5  
pos = 7

Insertion anywhere else:

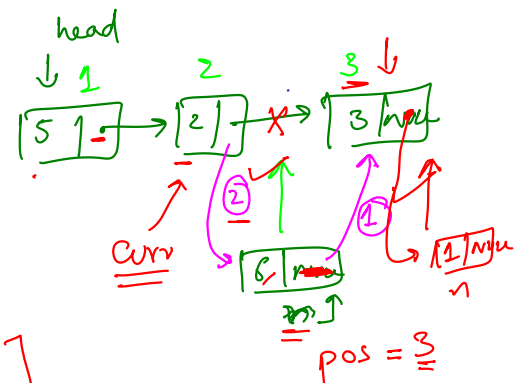
```

void insert(int val, int pos)
{
    if (pos < 1 || pos > size + 1)
    {
        cout << "Invalid pos";
        return;
    }
    node * n = new node(val);
    if (pos == 1) // Case 1:
    {
        n->next = head;
        head = n;
    }
}

```

insert(1, 3)

insert(1, 5)



```

}
else
{
    node * curr = head;
    for (int i = 1; i < (pos - 1); i++)
    {
        curr = curr->next;
    }
    n->next = curr->next;
    curr->next = n;
    size++;
}

```

insert(6, 3)

```

int search (int val)
{
    if (head == null)
    {
        return 0;
    }

```

return position

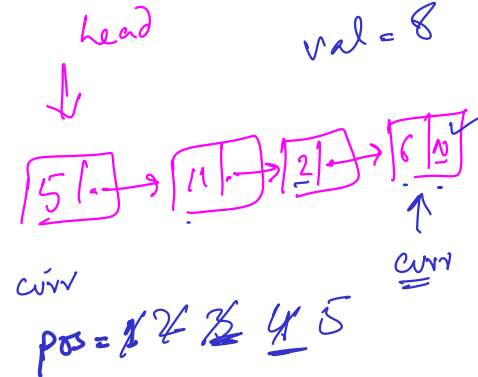
~~1~~ 0      1 5

```

    int pos = 1; ✓
    node *curr = head;
    while (curr != null)
    {
        if (curr->data == val)
        {
            return pos; ✓
        }
        curr = curr->next;
        pos++;
    }
    return 0;
}

```

Search(8)  
0 not found  
 size = 4  
 val = 8



```

void delete (int pos)
{
    if (pos < 1 || pos > size) ✓
    {
        return;
    }

```

```

    node * curr = head;
    if (pos == 1) // head
    {
        head = curr->next;
        delete curr;
    }

```

```

    else
    {
        for (int i = 1; i <= (pos-1); i++)
            curr = curr->next;

        node * delptr = curr->next;
        curr->next = delptr->next;
        delete delptr;
    }

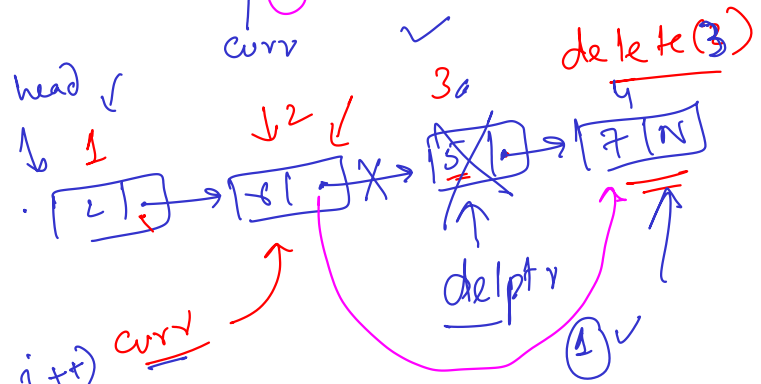
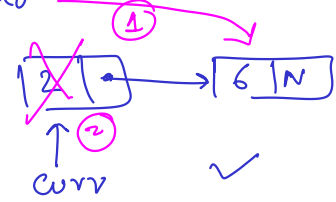
    size--;
}

```

size = 5      pos = 1 - 5

size = 0

head delete(1) ✓



delete(7)