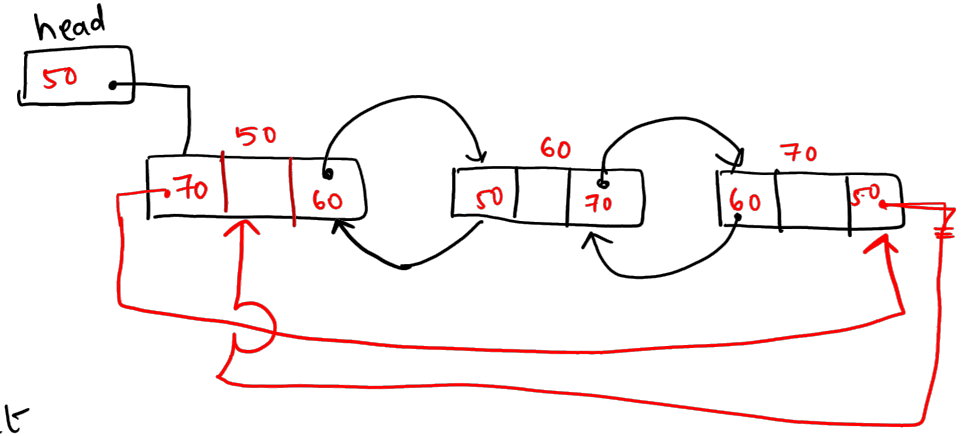
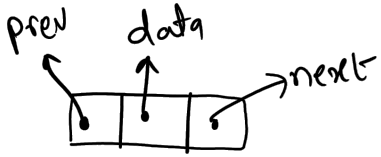
Circular Double Linked list:-






```
Struct Node
{
    int data;
    Struct Node *prev;
    Struct Node *next;
};
```

Create head:-

```
Struct Node *head = NULL;
```

Create a node:-

```
Struct Node *nn = (Struct Node *) malloc ( size of (Struct Node))
```

Operations:-

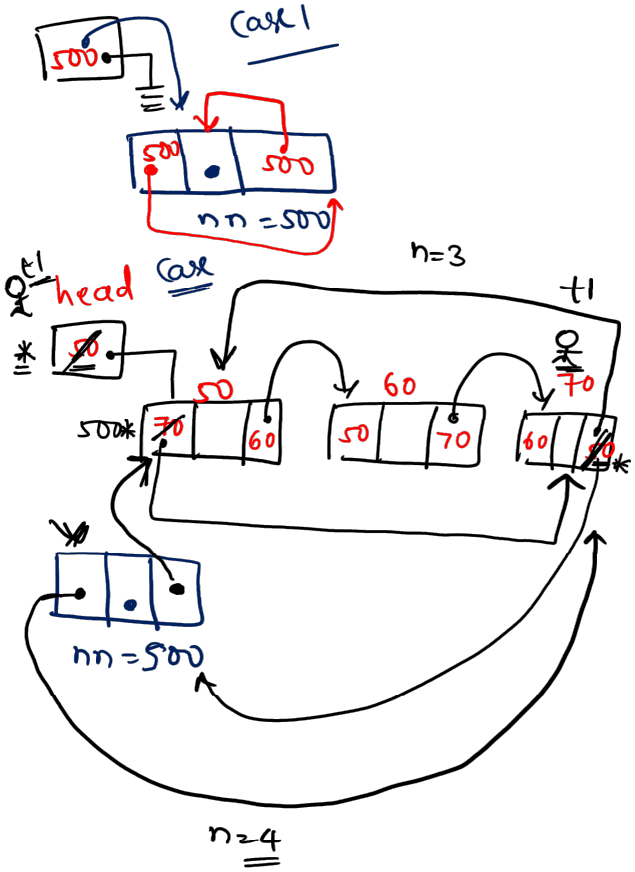
(i) insertion
- at begin
- at end
- at given position

(ii) deletion
- at begin
- at end
- at given position

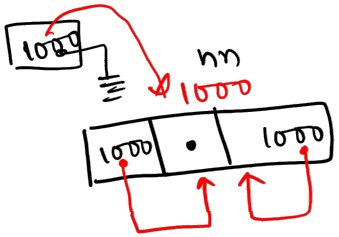
(iii) traversing / display

(iv) searching

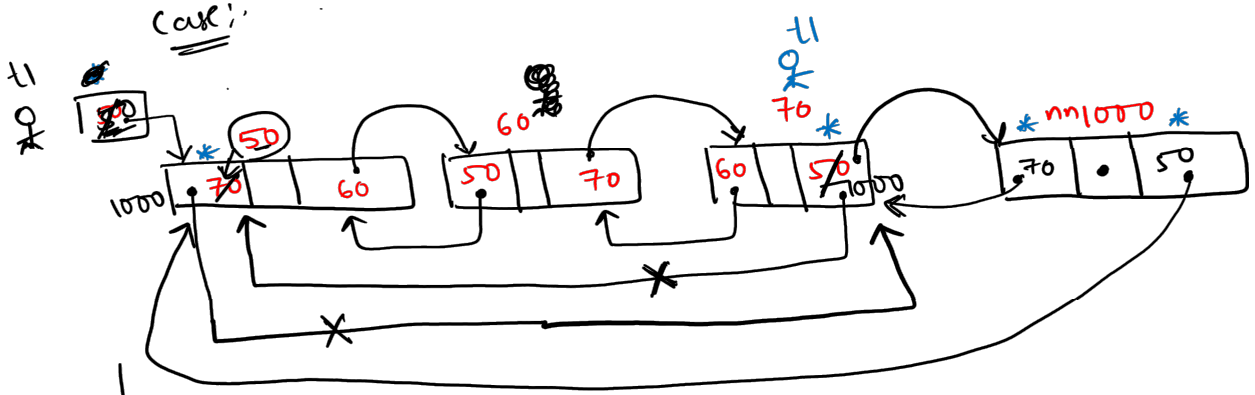
```
void insertatBegin ( int item)
{
struct Node *nn = (struct Node *) malloc( sizeof (struct Node));
   nn -> data = item;

   if (head == NULL)
      {
          nn -> next = nn;
          nn -> prev = nn;
           head = nn;
      }
      else
       {  struct Node *tl = head;
          while (tl -> next != head)
             {   tl = tl -> next;
             }
          tl -> next = nn;
          nn -> next = head;
          head -> prev = nn;
          nn -> prev = tl;
          head = nn;
       }
       n++;
}
```



Case 1:



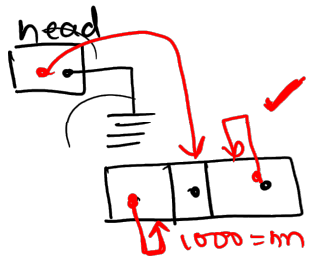
Case 2:

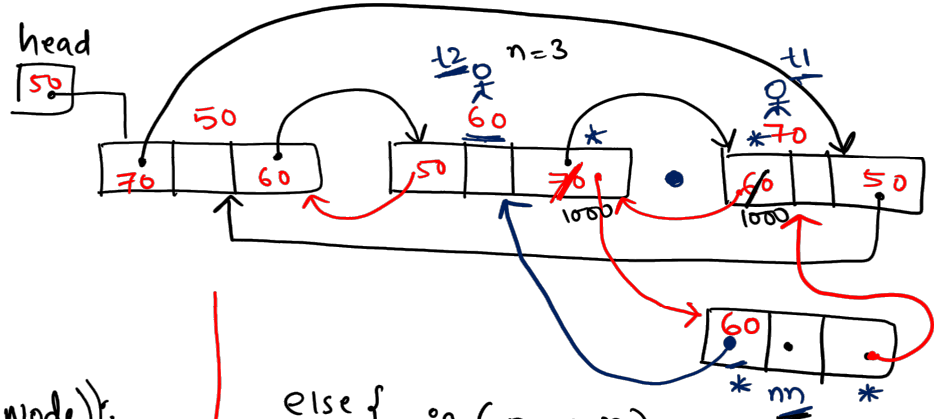



```
Void insert at End (int *item);
{
 SN *nn = (SN*) malloc(sizeof(SN));
   nn->data = *item;
   if (head == NULL)
   {
       nn->next = nn;
       nn->prev = nn;
       head = nn;
   }
```

```
else
{  struct Node *tl = head;
   while( tl->next != head)
   {   tl = tl->next;
   }
   tl->next = nn;
   nn->prev = tl;
    nn->next = head
     head->prev = nn;
}
```

insert at given position



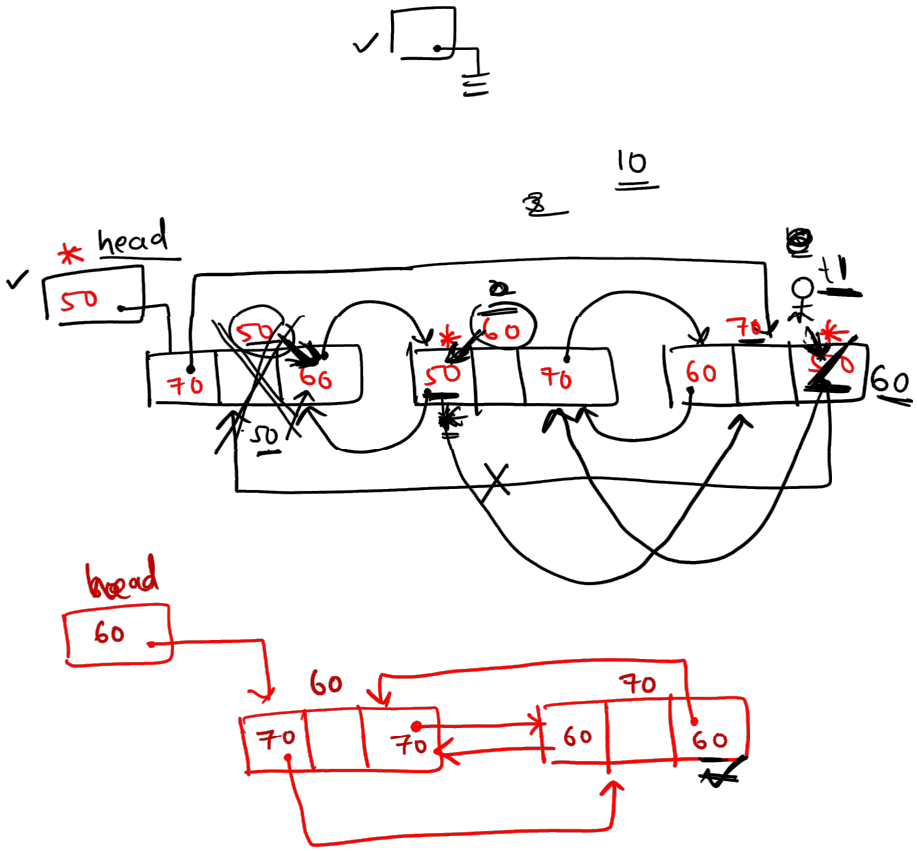


pos 10

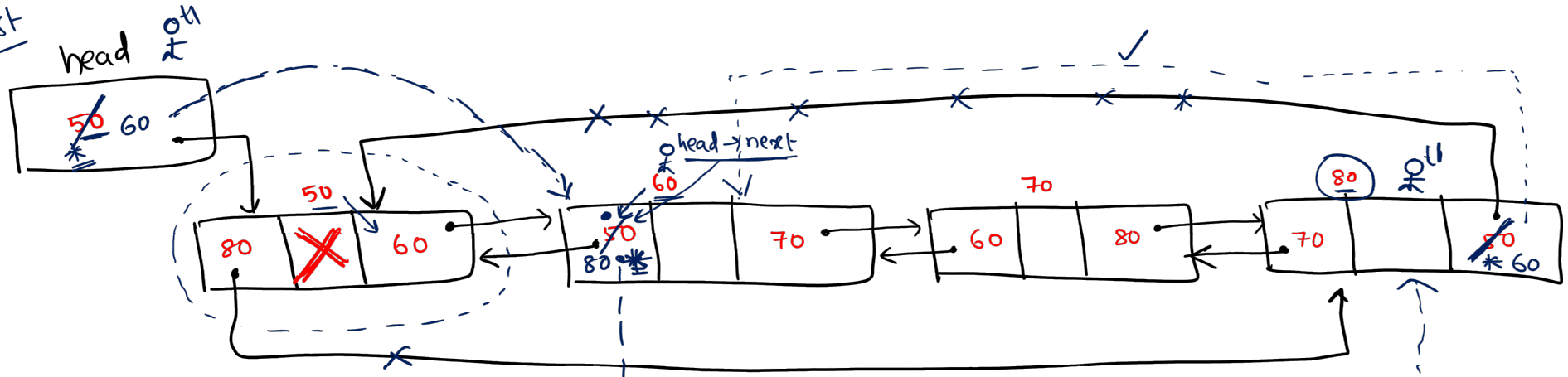
```
Void insertatgivenposition (int item, int pos)
{
  SN *nn = (SN *) malloc( sizeof (struct Node));
  nn -> data = item;
  if ( head == NULL)
  { printf(" list is empty - so iam addidng at 1st position");
     nn -> next = nn;
     nn -> prev = nn;
      head = nn;
  }
  else {  if (pos > n)
          { printf(" invalid position")
          }
          else{ struct Node *t1 = head;
                struct Node *t2;
          for(i=1;i<pos;i++)
          { t2 = t1;
            t1 = t1->next;
          }
          nn -> prev = t2;
          nn -> next = t1
          t2 -> next = nn;
          t1 -> prev = nn;
          }
          }
```

Deletion:-

```
void deleteatbegin( )
{
  if(head == NULL)
  {
     printf(" Lis is empty → No deletion");
  }
  else{
     struct Node *tl = head;
      while(tl→next!=head)
      {
         tl=tl→next;
      }
        tl→next = head→next;
        (head→next) → prev = tl;
```



Delet at first



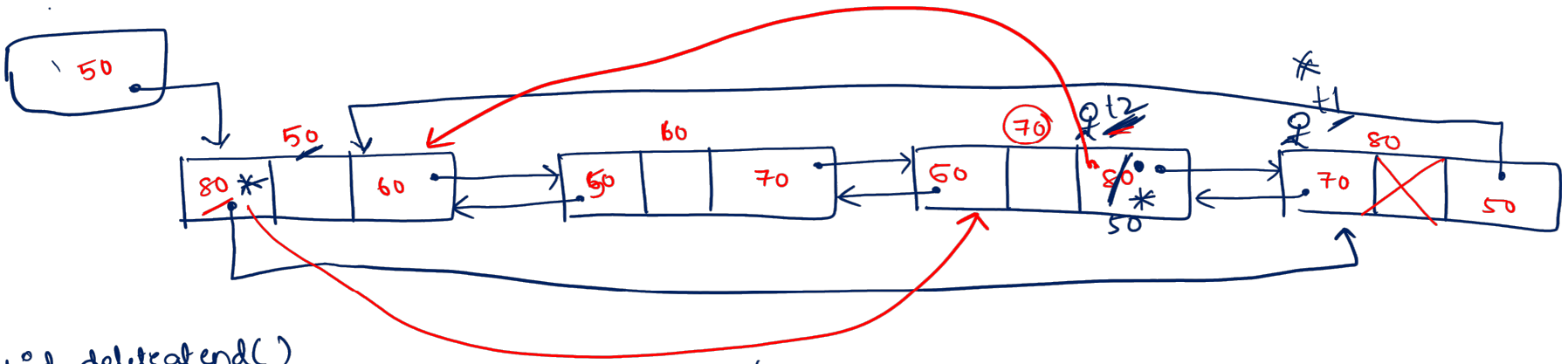

```
else{
    Struct Node *tl = head;
    While (tl->next != head)
    {
      tl = tl-> next;
    }
     tl->next = head->next;
    (head->next) -> prev = tl;
        head = head->next;
}
```

head->next

60 ⟶ prev = tl

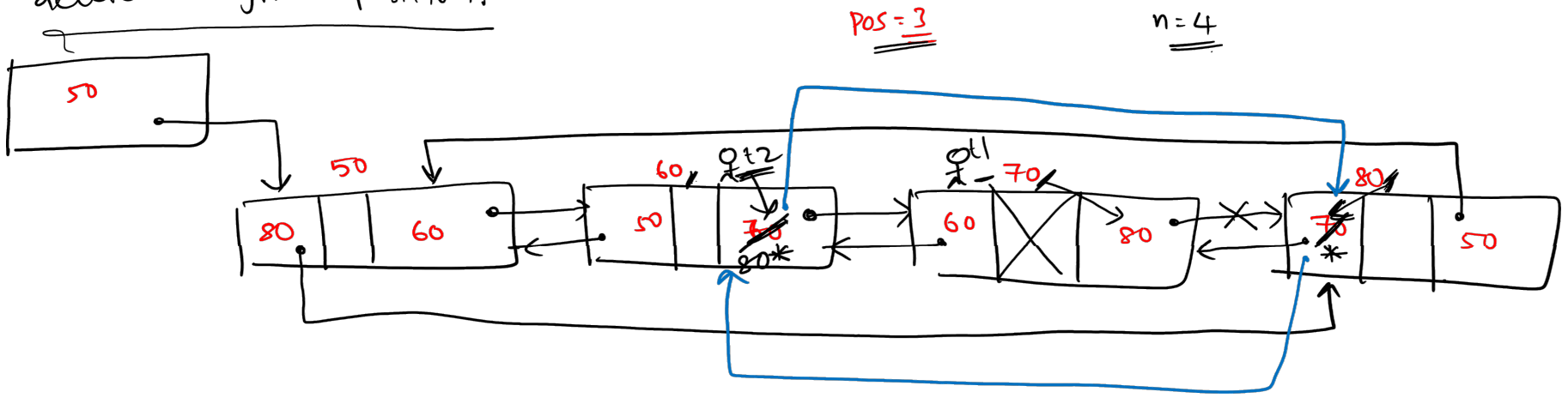
↓

(head->next) -> prev = tl




```
void deleteatend()
{
    if(head==NULL)
    {
        printf("List is empty");
    }
    else{
        struct Node *t1=head;
        struct Node *t2;
        while(t1->next!=head)
        {
            t2=t1;
            t1=t1->next;
        }
        t2->next=head
        head->prev= t2;
    }
    n--;
}
```

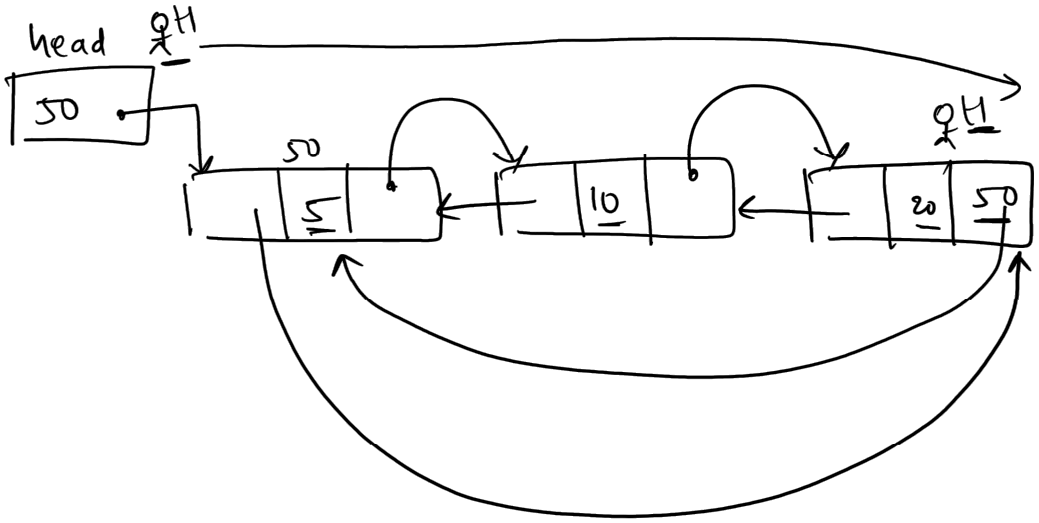
delete at given position:-



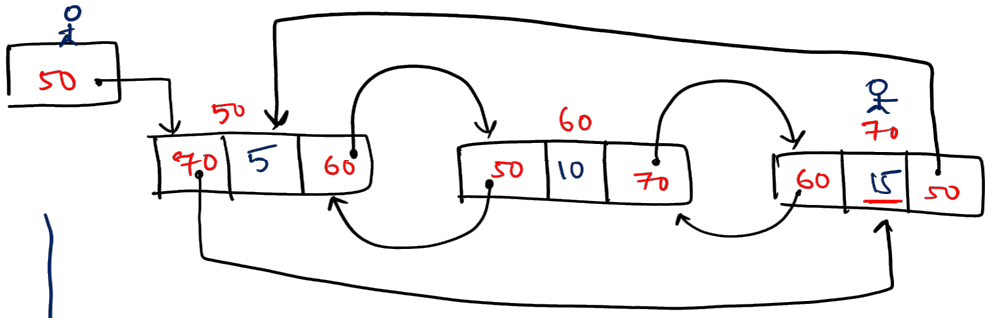

```
void deleteatgivenpos (int pos)
{
  if (head==NULL)
  {
    print("List is empty - no delete");
  }
  else {
    if( pos > n)
    {
      printf(" invalid position");
    }
  else {
    struct Node *t1=head;
    struct Node *t2;
    for(i=1; i<pos; i++)
    {
      t2=t1;
      t1=t1->next;
    }
    t2->next=t1->next;
    (t1->next)->prev = t2;
    n--;
  }
}
}
```

traversing / display :-

```
void display()
{
   if(head == NULL)
   {
     printf(" List is empty ");
   }
   else{  struct Node *tl=head;
        while (tl->next!=head)
         {
            print(" %d", tl->data);
              tl = tl->next;
         }
}
```



Seaching ;.



se = 15

```
void searching(int se)
{  int pos = 0;
   int status = 0;

  Struct Node *tl = head;

while(tl->next != head)
 {
    if (tl->data == se)
    {
     print("element found at %d", pos);
     status = 1;
    }
     pos++;
}
```

```
if(status == 0)
{
  printf("element doesnot found");
}
}
```

Status = 0

Status = 1