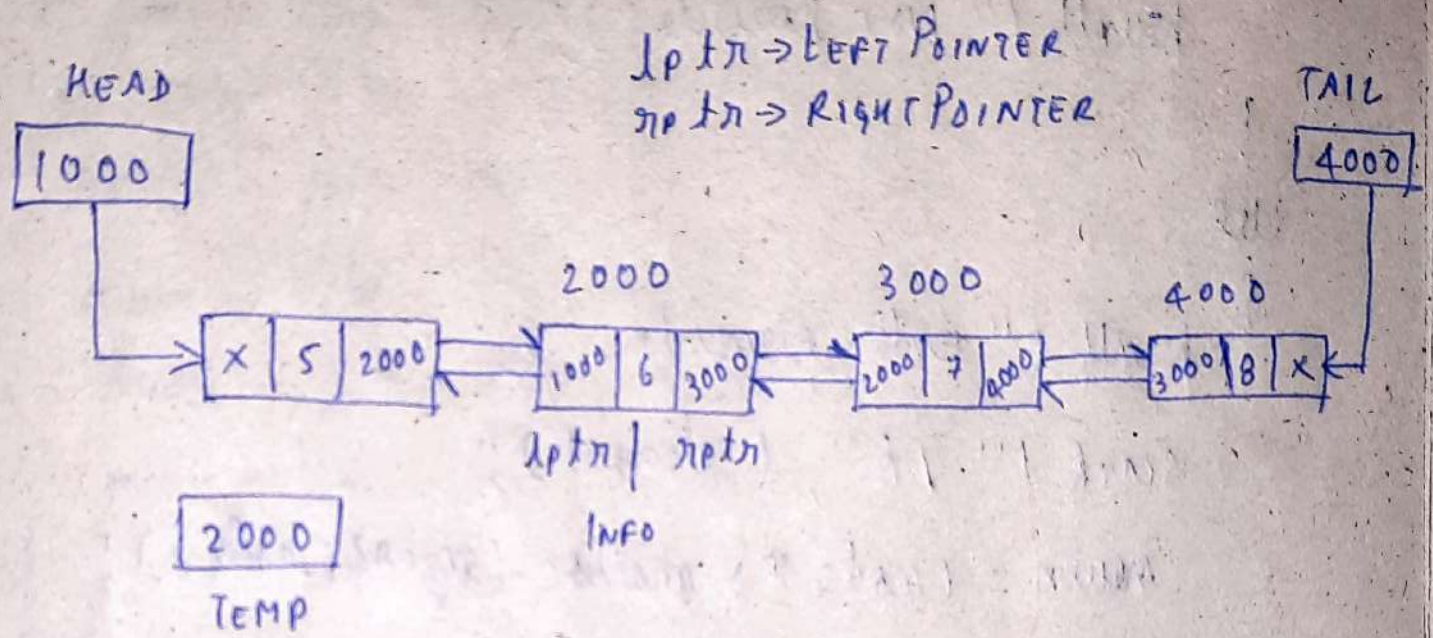
02-08-2024

~~DOUBLE LINKED LIST~~

~~DOUBLE ENDED~~

DOUBLE LINKED LIST =>

lptr → LEFT POINTER
rptr → RIGHT POINTER



if (head == NULL)
// empty

if (index == 0)
// list first 1st

index = size
&
2 A > T

if (index == size)
// list presentlast

Double
linked
list

PPS

DS

```
-> # include <stdio.h>
   # include <stdlib.h>
   struct Node
   {
     struct node *Lptr;          // Lptr -> LEFT POINTER
     int info;
     struct node *Rptr;          // Rptr -> RIGHT POINTER
   };
   typedef struct node node;
   node *head = NULL, *tail = NULL;
   node *temp, *newn;            // newn -> NEW NODE
   int size = 0;
   void insertion ( )
     { int index, element;
```

```
printf ("Enter index to insert : ");
scanf ("%d", &index);
if (index < 0 || index > size)
{
    printf ("Not possible");
}
else
{
    printf ("Enter elements : ");
    scanf ("%d", &element);
    newn = (node *) malloc (sizeof(node));
    newn -> info = element;
    if (head == NULL)    // Empty
    {
        newn -> lptr = newn -> rptr = NULL;
        head = tail = newn;
    }
    else if (index == 0)    //List present at 1st
    {
        newn -> lptr = NULL;
        newn -> Rptr = head;
        head = head -> lptr = newn;
    }
    else if (index == size)    //Last : List present at last
    {
        newn -> Rptr = NULL;
        newn -> lptr = tail;
        tail = tail -> rptr = newn;
```

```
else
{
  temp = head;
  for (i = 1; i < index; i++)
  {
    temp = temp -> Rptr;
  }
  newn -> Lptr = temp;
  newn -> Rptr = temp -> Rptr;
  newn -> Lptr -> Rptr = newn;
  newn -> Rptr -> Lptr = newn;
 }
 size++;
 }
}
void display1()
{
 temp = head;
 while (temp != NULL)
 {
  printf("%d", temp -> info);
  temp = temp -> Rptr;
 }
}
void display2()
{
 temp = head;
 while (temp != NULL)
 {
  printf("%d", temp -> info);
  temp = temp -> Lptr;
 }
}
```

DOUBLE LINKED LIST =>

```
-> #include<stdio.h>
   #include<stdlib.h>
   struct node
   {
     struct node *lptr;      //lptr - Left pointer
     int info;
     struct node *rptr;      //rptr - Right pointer
   };
   typedef struct node node;
   node *head = NULL, *tail = NULL;
   node *temp, *newn;      //newn - new node
   int size = 0;
   void insertion()
   {
     int index, element, i;
     printf("Enter index to insert : ");
     scanf("%d", &index);
     if (index < 0 || index > size)
     {
       printf("NOT POSSIBLE \n");
       return;
     }
     else
     {
```

```
printf ("Enter element: ");
scanf ("%d", &element);
newn = (node *) malloc (sizeof (node));
newn -> info = element;
if (head == NULL)          // EMPTY List
{
    newn -> lptr = newn -> rptr = NULL;
    head = tail = newn;
}
else if (index == 0)       // Insert at the begining
{
    newn -> lptr = NULL;
    newn -> rptr = head;
    head -> lptr = newn;
    head = newn;
}
else if (index == size)    // Insert at the end
{
    newn -> rptr = NULL;
    newn -> lptr = tail;
    tail -> rptr = newn;
    tail = newn;
}
else
{
    temp = head;           // Insert in the middle
```

```
        for(i=0; i<index-1; i++)
        {
            temp = temp->rptr;
        }
        neww->lptr = temp;
        neww->rptr = temp->rptr;
        temp->rptr->lptr = neww;
        temp->rptr = neww;
    }
    size++;
  }
}

void display1()
{
    temp = head;
    while (temp != NULL)
    {
        printf("%d ", temp->info);
        temp = temp->rptr;
    }
    printf("\n");
}
void display2()
{
    temp = tail;
    while (temp != NULL)
    {
```

```
        printf ("%d ", temp -> info);
        temp = temp -> lptr;
    }
    printf ("\n");
}
int main ()
{
    int ch;
    while (1)
    {
        printf ("Enter \n 1 to insert \n 2 to display from head to tail \n 3 to display from tail to head \n 4 to EXIT \n ");
        scanf ("%d", &ch);
        switch (ch)
        {
            case 1: insertion (); break;
            case 2: display1 (); break;
            case 3: display2 (); break;
            case 4: exit (0); break;
            default: printf ("INVALID CHOICE \n");
        }
    }
    return 0;
}
```