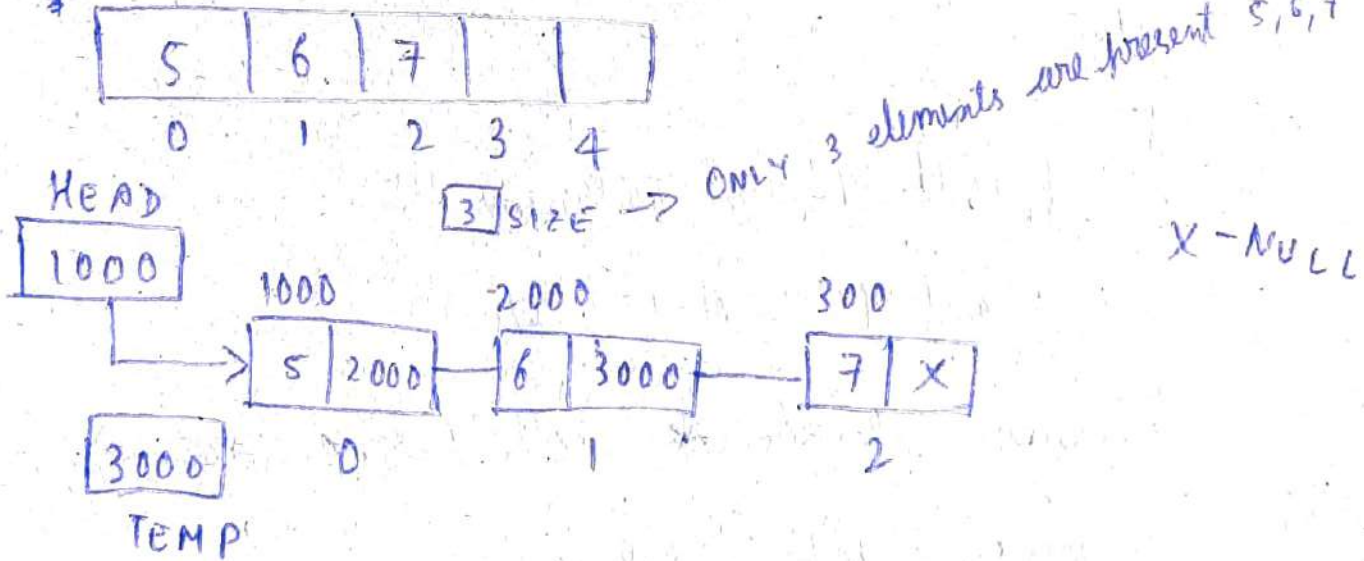
31-07-2024

LINKED LIST =>



-> Struct node

SINGLY LINKED LIST WITH INSERTION, DELETION & DISPLAY ()

```
{
  int info;
  struct node *ptr;
};
typedef struct node node;
node *head = NULL, *temp, *newn;
int size = 0;
void insertion ()
 {
   int i, index, ele;
   printf ("Enter index to insert ");
   scanf ("%d", &index);
   if (index < 0 || index > size)
   {
     printf ("not possible ");
   }
   else
   {
     printf ("Enter element ");
     scanf ("%d", &ele);
   newn = (node *) malloc (sizeof (node));
     newn -> info = ele;
     if (index == 0)
     {
       newn -> ptr = head;
           head = newn;
     }
```

newn -> new node

✓ insertion
deletion
display
exit

```
else
{
    Temp = head;
    for (i = 1; i < index; i++)
        temp = temp -> ptr;
    newn -> ptr = temp -> ptr;
    temp -> ptr = newn;
  }
  size++;
}
}
```

newn -> new node

```
void display ()
{
  temp = head;
  while (temp != NULL)
  {
    printf ("%d", temp -> info);
    temp = temp -> ptr;
  }
}
```

temp -> info => Value
temp -> ptr => Value of address

SINGLY LINKED LIST WITH INSERTION DELETION & DISPLAY ( ) ;-

```
-> #include <stdio.h>
   #include <stdlib.h>
   struct node
   {
     int info;
     struct node *ptr;
   };
   typedef struct node node;
   node *head = NULL, *temp, *newn;   // newn - new node
   int size = 0;
   void insertion ()
   {
     int i, index, element;
     printf ("Enter index to insert: ");
     scanf ("%d", &index);
     if (index < 0 || index > size)
     {
       printf ("Not possible \n");
     }
     else
     {
       printf ("Enter
       newn = (node*) malloc (sizeof (node));
         // allocate memory for new node
       printf ("Enter element: ");
       scanf ("%d", &element);
```

```
        neww -> info = element;   // neww - New node
        if (index == 0)
        {
            neww -> ptr = head;   // neww - New node
            head = neww;
        }
        else
        {
            temp = head;
            for(i = 1; i < index; i++)
            {
                temp = temp -> ptr;
            }
            neww -> ptr = temp -> ptr;
            temp -> ptr = neww;
        }
        size++;
    }
}

void deletion ()
{
    int i, index;
    printf ("Enter index to delete: ");
    scanf ("%d", &index);
    if (index < 0 || index >= size)
    {
        printf ("Not possible \n");
    }
```

```
else
{
    node *del;
    if (index == 0)
    {
        del = head;
        head = head -> ptr;
    }
    else
    {
        temp = head;
        for (i=1; i<index; i++)
        {
            temp = temp -> ptr;
        }
        del = temp -> ptr;
        temp -> ptr = del -> ptr;
    }
    size--;
  }
}
void display ()
{
    temp = head;
    while (temp != NULL)
    {
        printf ("%d ", temp -> info);   // temp -> info = STORES VALUE
        temp = temp -> ptr;             // temp -> ptr = STORES VALUE OF ADDRESS
    }
    printf ("\n");
}
```

```
int main ( )
{
  int ch;
  switch
  while (1)
  {
    printf ("Enter \n 1 to insert \n 2 to delete
    \n. 3 to display \n 4 to exit \n ");
    scanf ("%d", & ch);
    switch (ch)
    {
      case 1: insertion ();
      break;
      case 2: deletion ();
      break;
      case 3: display ();
      break;
      case 4: exit (0);
      default: printf ("INVALID CHOICE \n");
    }
  }
}
```