union node (In union m/m is created as much as the large one)

```
{
member1  int a[5];      —— 10
member2  float b[20];   —— 80
member3  char c[50];    —— 50
};                      80 bytes
```

This is shared by all the ~~promise~~ members (one member is used at a time)

- Previous content is overwritten.
- In union M/m Sharing is present.

V.Imp

LINKED LIST (Interview)

☆ Struct node (6bytes)

```
{
    int data;          // 2bytes
    Struct node *next;  (pointer) → 4bytes
};
```

means address which is pointing to same kind of data type (i.e. Struct node) // Self Referential data structure

printf("%d", sizeof(structnode)) = 6bytes

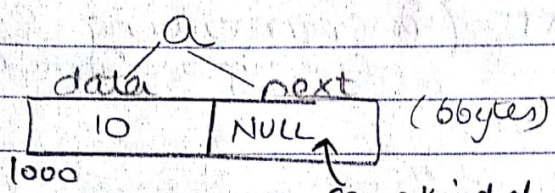
☆ Struct node

```
{
    int data;
    struct node next;   // wrong
};
```

It is not a pointer

Error (Not a self Referential data structure)

Scanned by CamScanner

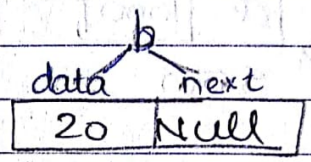


same kind of & data type present in it (of structnode type)
that's why we write struct node* Next

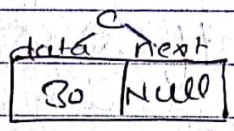
object created

Structnode a={10, Null}

Structnode b={20, Null}

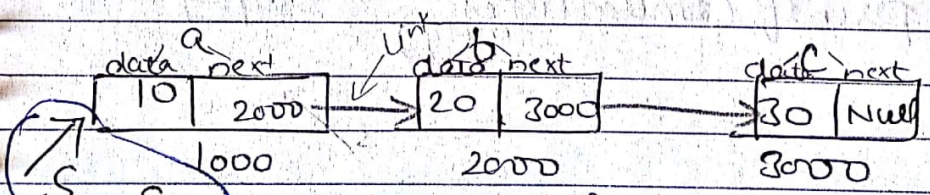


Struct node c={30, Null}



Linking (statically) (we are ourself specifying what shud we put in next)

a.next = &b;
b.next = &c;



S = S 1000 Structnode *s = &a;

List ⇒ means Collection / Structure of data.

printf(s) = 1000

(*s).data = 10 ≡ s→data = 10

(*s).next = 2000 ≡ s→next = 2000

(*s) 1000 inside 1000

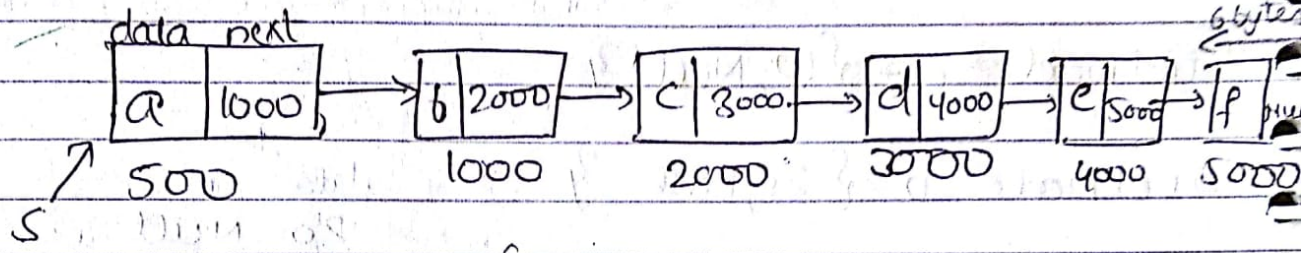
dynamic M/m allocation

malloc(6) = allocate M/m of 6 bytes

free(1000) = free M/m of 1000 bytes

★ S = S → next (incrementing S)
(pointing everytime other node.

Ques: Consider the following Linked List

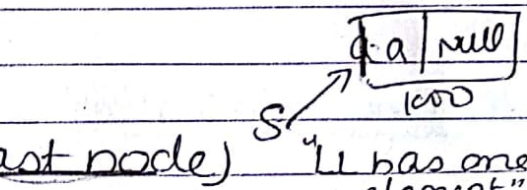


S
500

★ printf (S) = 500

Conditions (Important)

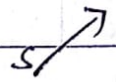
★ if (S → next == NULL)
printf (S is last node)



"LL has one element"

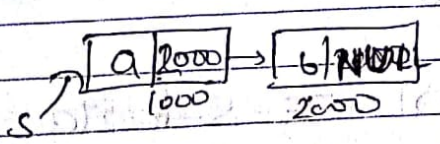
★ if (S == NULL)
printf (Linked List is empty)

Linked List is empty



★ if (S → next → next == NULL)
printf (LL has 2 elements)

LL has 2 elements



★ If S = NULL
then *S = *(NULL)
|| you can't go inside Null

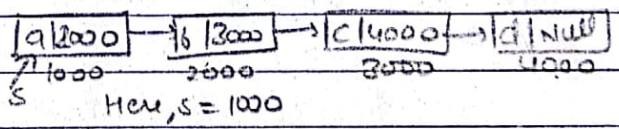
Segmentation Error
(Accessing some M/m location which doesn't exist)

Ques: Write a C program to print data part of every node in the given linked list.

S is a address of structnode type

```
PDL ( structnode *S )
(starting node)
{
    if(S==NULL) return ; //LL over
    else
    {
        while(S!=NULL)
        {
            printf(S->data);
            S=S->next;
        }
    }
}
```

Time complexity ⇒ O(n)



output: a b c d e f

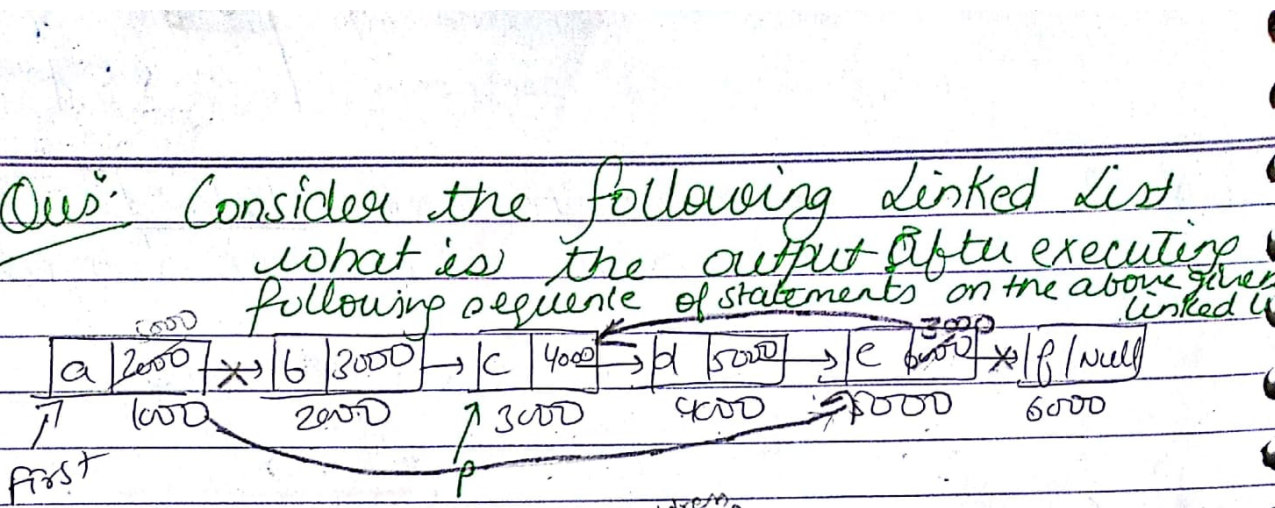
Ques: Write a C program to find the last node of the given linked list. O(n)

```
structnode* fLL (struct node *S)
{
    if (S==NULL)   //L.L. Over
        return;
    else
    {
        while (S->next!=NULL)
        {
            S=S->next;
        }
        ~~printf(S->data);~~ return (S);
    }
}
```

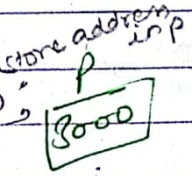
fLL finish return structnode type value

address

Ques: Consider the following Linked List what is the output after executing following sequence of statements on the above given linked list



(i) Struct node *p; store address in p



(ii) p = first → next → next;

(iii) first → next = p → next → next;

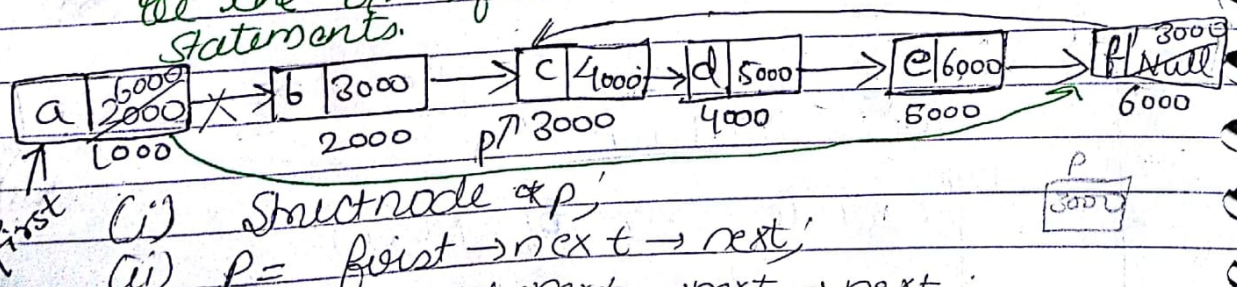
(iv) p → next → next → next = p

(v) printf (first → next → next → next → data);

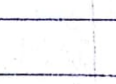
d

Ans: d

Ques: Consider the following Linked List. what will be the o/p if we execute the following statements.



(i) Structnode *p;



(ii) p = first → next → next;

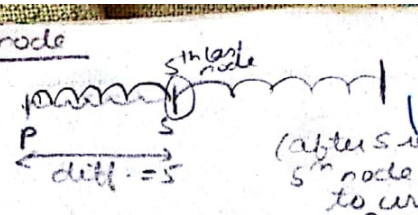
(iii) first → next = p → next → next → next;

(iv) p → next → next → next = p;

(v) printf (first → next → next → next → next → data);

Ans: e

Scanned by CamScanner



156

(after S reached 5th node say P to work 2 from ~~P & S~~ ~~P & S~~ ~~P & S~~

else

2. if (s → next = = NULL) return;

3. while (s → next → next != NULL)

```
{
    s = s → next;
}
return (s);
```

- Most of the problems in single linked list can be solved by the friend node

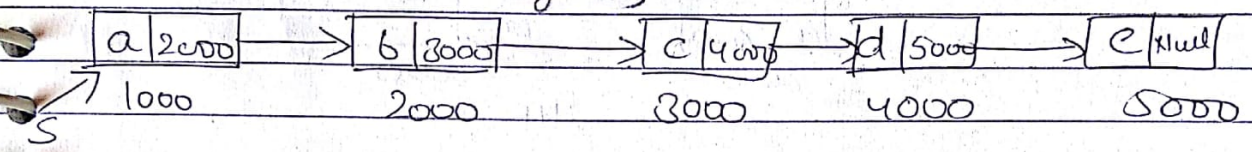
Scanned by CamScanner

5th/June/2013

155

Lecture 17

Ques: Write a C program to Insert a node with data x at the end of the given Linked List.
↳ (starting address given)




```
Structnode   (16 bytes)
{
    int data;   //2 bytes
    Structnode *next;   //4 bytes
}
```

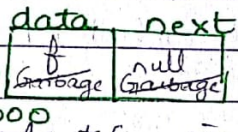
↑ address ↑ address of (same structnode type)

// Structnode *s = 1000; (s is 1000)
*s = Inside the node

(those m/m which are created dynamically there can only deallocate dynamically (using free))

Func. returning address ↓

Structnode* ← This func. is finally returns address of structnode type p



data for node

```
Insertatend (Structnode *s, int x)
{
    Structnode *p, *q;   q = s;
    p = (Structnode *) malloc (sizeof(Structnode));
    (*p).data = x;   (*p).next = null;
```

p: 6000 (contain 6000 structnode pointer)

Type Casting (pointer)

(p is 6000)
(*p = Inside p)

Getnode(x);

1. If (s == null) s ↗ (or) NULL s
{
    s = p;
    return s; //returning the address of s node
}

create the node with data x & return the address where it is created.

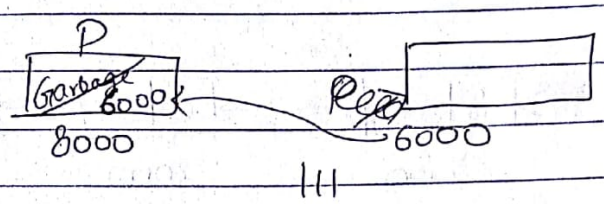
Continue on the next to next page

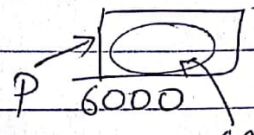
Explanations :→

```
structnode *p;
p=malloc(6Bytes)
```

= Somewhere 6 Bytes M/m will be created for you. (The address of that m/m is given to us).

(Struct node pointer) (Type Conversion) 6000 (void pointer)





(p is a pointer pointing to 6000)

6Bytes data is stored here (Struct node type data is stored).

$$p = \underbrace{(\text{Structnode } *)}_{\text{Type Casting}} \underbrace{\text{malloc}(\text{6 Bytes})}_{\text{void pointer}}$$

Structnode node pointer (6000)

6000 — sizeof(Struct

```
p= (structnode*) malloc(sizeof(structnode));
```

"Creation of M/m"

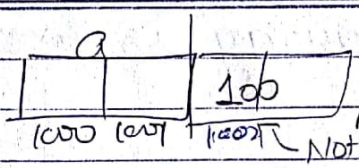
Note :- ☆ If M/m is not allocated to you (you can only read, not write in the Memory)

☆ If M/m is allocated to you (you can read & write in it).

Scanned by CamScanner

* Null = Segmentation Error (as we can't go inside the non-available M/m)
Non available M/m

int a=10;
printf(a) => 10



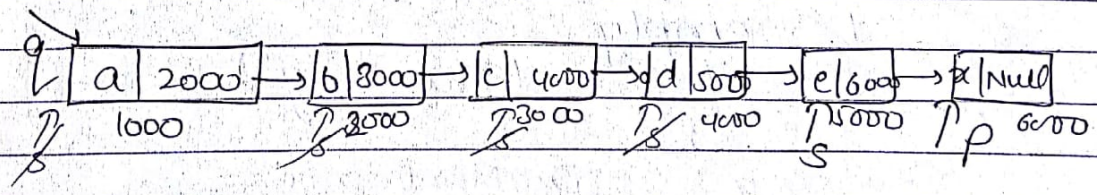
*(&a+1) = 100;
1000
1002

printf(*(&a+1)) => Garbage (∵ You can't write in the M/m not related to you)

2. while(S->Next != Null)
↳ this node is not the last node
// L.L is not empty, there is atleast one element

{
S = S -> Next;
} // coming out of while loop means you are at the last node

3. S->next = p;
return (q); → return the starting address of the Linked List.
}



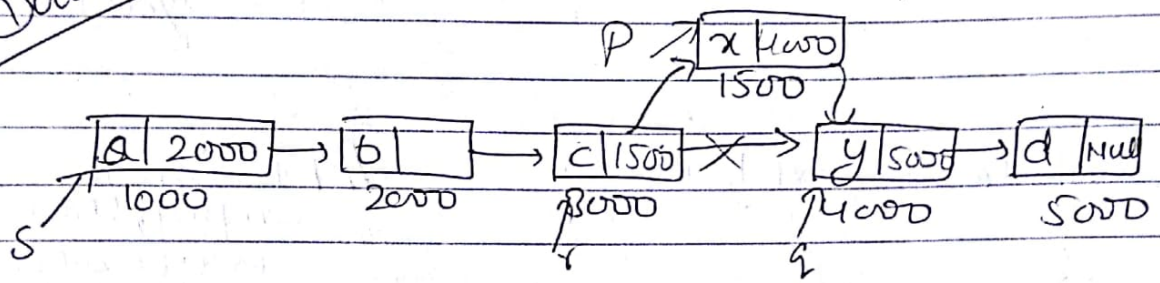
(int*) f (int*, int*); // returns Integer pointer as O/P.
2 Integer pointers as I/P

(int *) f (float *, (char *) g (int, int *))
↓ returns Integer pointer O/P.
↓ func which will takes 2 parameters as I/P. 1st is float pointer & 2nd parameter is character pointer.
float pointer
character pointer
↓ Integer
↳ Integer pointer

• Size of array is fixed (you can't Insert any element in it)

Question Write a C program to insert a node with data 'x' ~~before~~ 'x' before a node with data 'y'.
(use friend concept)

Solution:



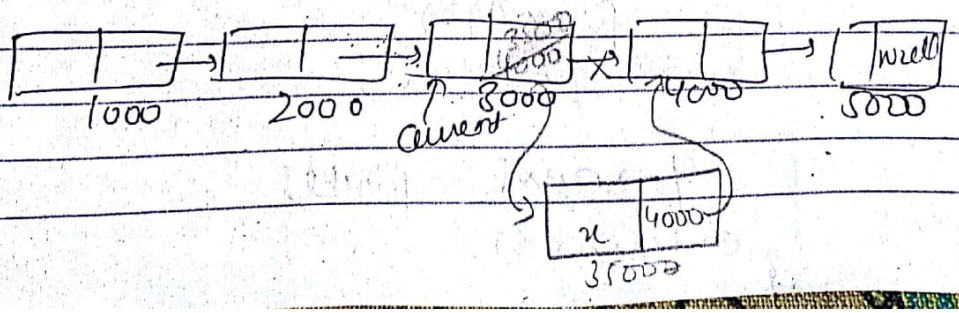

```
Structnode* Insertxbeforey(int x, Structnode *s, int y)
{
    Structnode *p;
    p=(Structnode*) malloc(sizeof(Structnode));
      p->data = x;
      p->Next = null;
      1. empty (I can't do anything if L.L is empty)
(only 1 node in L.L with data 'y')  2. if (s->data == 'y') (First node itself contains y)
      {
          p->Next = s;
          s=p; return(s);
      }

3.    q=s, r=Null;   (my friend)

      while(q->data != y && q->next != null)
      {
          (current data is not my data)   (next node is available)
```

```
        r = q; // put the friend before me
        q = q -> next;
    } //why I came out of the for loop
    if ( q -> data == 'y') //if this will fail
    {                       means L.L. is over
        r -> next = p;      & 'y' is not there
        p -> next = q;      in list so we
                            don't add 'x'
    } else { printf("L.L. is over, y is not there
                so don't add x"); }
    return(s);
}
```

. Time Complexity = O(n). (Even then O(n) is not at all possible ∵ random access is not allowed).

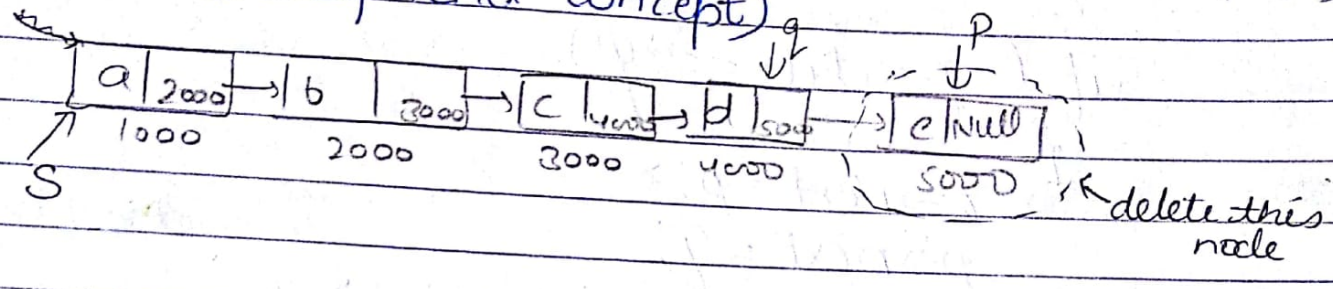
Question Which one of the following statement Inserts an element x after position current?

a) current = newnode(x, current) — create node with specified parameter
b) current = newnode(x, current->next);
c) current -> next = newnode(x, current);
d) current -> next = newnode(x, current -> next);

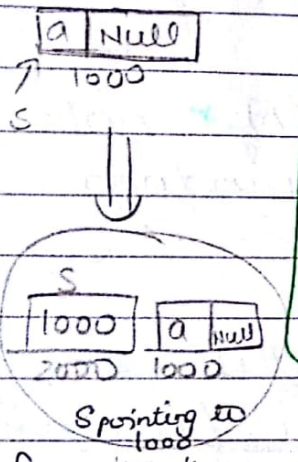


Ques: Write a C program to delete a node at the end of given linked list. (use free also to deallocate the M/m) (use friend Concept)

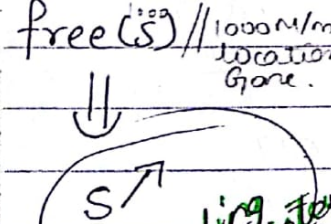
remove the link




```
Structnode * DeleteLast(Structnode * S)
{
    1. empty L.L. (don't do anything)
    2. if(S->next == Null) (exactly 1 node)
    {
        free(S);
        S = Null;
    }
    3. p = S, q = Null;   // My friend.
    while(p->next != Null) // You are not last node
    {
        q = p;
        p = p->next;
    }
    free(p);
    p = Null;   // Now, p is not a dangling pointer
    q->next = Null;
    return(S);
}
```



free(S) // 1000 M/m location is Gone.

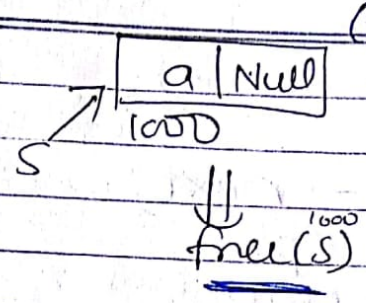


(M/m which is not now yours but previously yours)

S is pointing to someone which is not present. (dancing in the air). It will give wrong answer but not gives Segmentation Error.

Scanned by CamScanner

Ex.

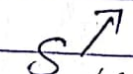


(S is pointing to 1000 which is allocated)

(*S).data = a

printf (S) = 1000 (1000 which is allocated to me)

(dangling pointer)



(S is still pointing 1000 but 1000th/m location is not available)

(S is pointing to 1000 which is deallocated, so S is called dangling pointer)

printf (S) = 1000 (1000 which is not allocated to you)

(*S).data = Garbage.

→ dangling pointer gives wrong answer but not the Segmentation Error.

- we should write program in such a way that it doesn't create the dangling pointer.

Eg. 1. free(S);

← upto here S is dangling pointer

2. S = null ← this will remove the dangling pointer

Ques. Write a C program to delete a node with data x from the given linked list.

Solution

```
Structnode* Delete x( Structnode *S, int x)
{
    1. empty
```

2. if (S → data == x)

{

p = s;

S = S → next;

free(p);

p = Null;

}

p = s, q = null;
(my friend)

while (q → data != x && p → next != NULL)

{

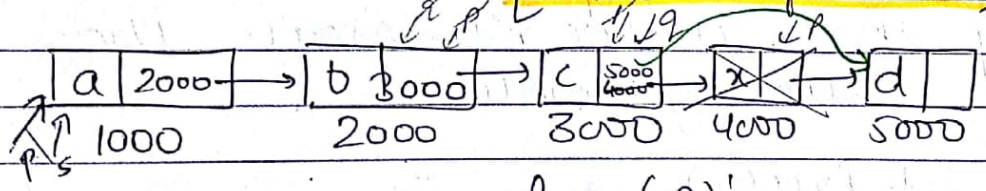
q = p;

p = p → next;

}

if (p → data == x)

{ q → next = p → next;



free(p);

p = Null;

}

else

{

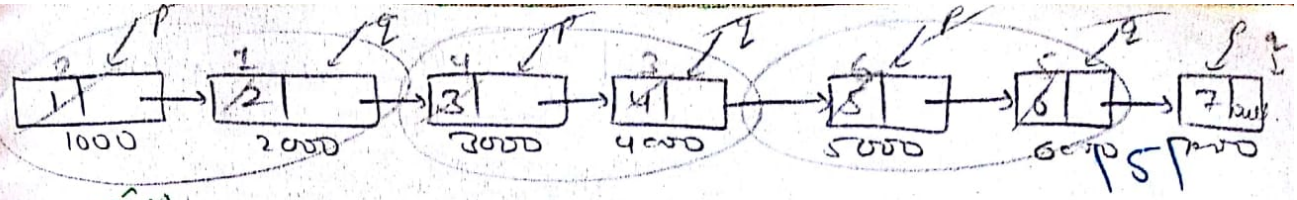
printf("No x found");

}

return (S);

}

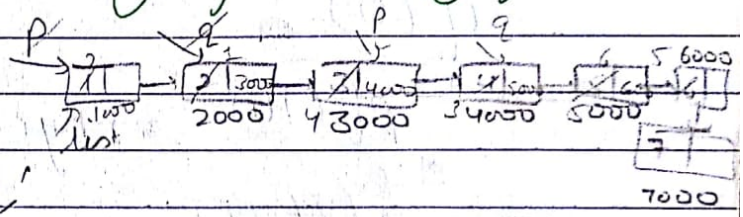
Scanned by CamScanner



Question:

The following C ~~program~~ func. takes single L.L. as a parameter & rearranges the elements of the list. The function is called with the list containing the integers 1, 2, 3, 4, 5, 6, 7 in given order, then what will be the contents of list after executing following func.

Solution:




```
Struct node
{
    int value;
    structnode *next;
};

void rearrange (struct node *list)
{
    Structnode *p, *q;
    int temp;

    if (!list || !(list->next)) return;
    if (list==Null)
    else // link list has min. 2 nodes.
    {
        p=list; q= list->next;
        while (q)
        {
            swap(p->value, q->value);
            p= q->next;
            q = p->next;
        }
    }
}
```

op: 2, 1, 4, 3, 6, 5, 7

if (NonZero) → always True
{ ≡ }

if (Zero) → always false
{ ≡ }

Note:

nonzero

```
if (1)
  printf(Hi)   ⇒ Hi
else
  pf(Bye)
```

Nonzero

```
if (10)
{ printf(Hi)   ⇒ Hi
}
else
  pf(Bye)
```

```
if (0)
{
  pf(Hi)   ⇒ Bye
else
  pf(Bye)
```

True = 1 Nonzero
False = Zero

x=10, y=0

```
if (x==y)
  pf(Hi)   ⇒ Bye
else
  pf(Bye)
```

x=10, y=20

```
if(x=y)
  pf(Hi)   ⇒ Hye
else
  pf(Bye)
```

x= 0, y=20

```
if (x=y)
  pf(Hi)   ⇒ Hi
else
  pf(Bye)
```

x= 10, y=0

```
if( x=y)
  pf(Hi) ⇒ Bye
else
  pf(Bye);
```

float ⇒ 0.0000---- 0
Char ⇒ '\0'
Integer ⇒ 0
⇒ Null
⇒ zeros

if (0) ⇒ fail

Scanned by CamScanner

- printf("%d", Null) ⇒ 0 (ASCII value of Null is 0).

$\boxed{\text{*Null = Segmentation Error}}$.

if(!Null) ⇒ True.
|||
if(1) or if(!0)

Question

Consider the function f defined below :→

```
Struct item
{   int data;
    Struct item *next;
};

int f (struct item *p)
{
    return ((p==null) || (p->next==null) ||
    ((p->data <= p->next->data) && f(p->next)));
}
```

(OR) (OR)

0 or 1 it contain (can be expression)

For given L.L. P the func^n f returns 1 iff

a) p contains 0 element or 1 element (also correct)

b) The elements in list P are sorted in non decreasing order. (Increasing + all same elements) ⇒ True for this also [10]→[10]→[10]

(Best answer (contains opposite also))

c) the elements in list P are sorted in non increasing order

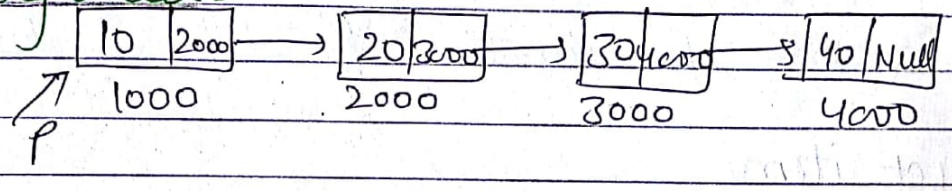
d) None

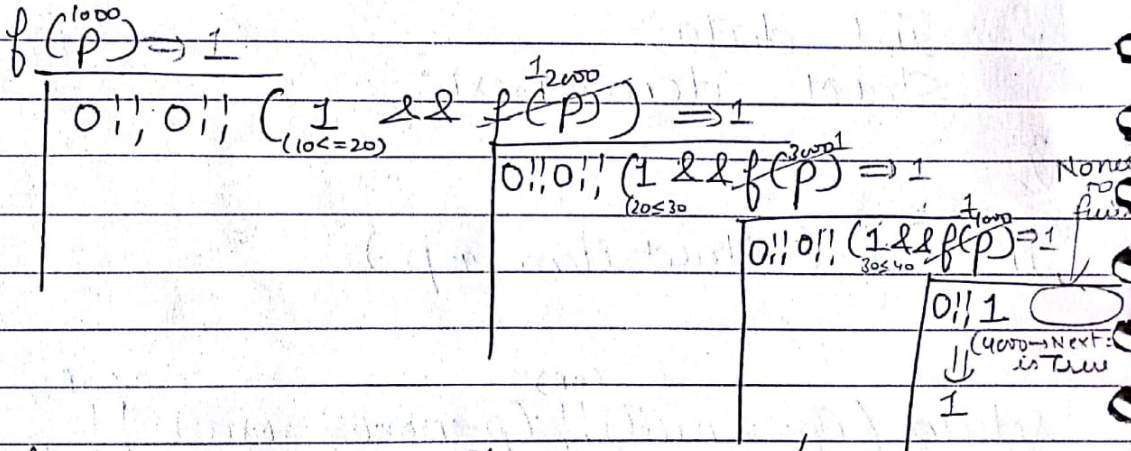
OR

if (a || b || c)

if a itself is true then don't goto b & c. But if a fails goto b & if a, b fails goto c, if all a, b & c fails means if (0).

Increasing order :-





Link List contain Zero element / one element it is still non-decreasing element. Option (b) contains option (a) also,

So, option (b) is Correct.

Scanned by CamScanner

Case:

```
struct item
{ int data;
  struct item *next;
};
```

```
int f(struct item *p)
{
return((p==Null)||(p->next==Null)||((p->data<=
  p->next->data)||f(p->next)));
}
```

replaced from &&

$f(p) \Rightarrow 1$ (p = 1000)

$0 || 0 || (0 || f(p))$ (p = 2000)

$0 || 0 || (0 || f(p))$ (p = 3000)

$0 || 0 || (0 || f(p))$ (p = 4000)

$0 || 1$

$\Rightarrow 1$

6 | 2000	→	5 | 3000	→	3 | 4000	→	1 | Null
1000		2000		3000		4000

↑ p

∴ This func. will return 1 for all non-decreasing or non-increasing elements.

Scanned by CamScanner

Question Following C program takes single L.L. as func. argument. It modifies list by moving last element to front of list & returns modified list. Some part of the code (logically deletion occurs moving last node to 1st node)

Note: Typedef Struct node
{
int value;
struct node *next;
}b;

Struct node a;
b a;
alias name of struct node

Note: By Typedef we have made the alias name of struct node as 'b'. Through 'b' we can create its object

Typedef struct node
{ int value;
struct node *next;
}node;
node is the alias name given to struct node

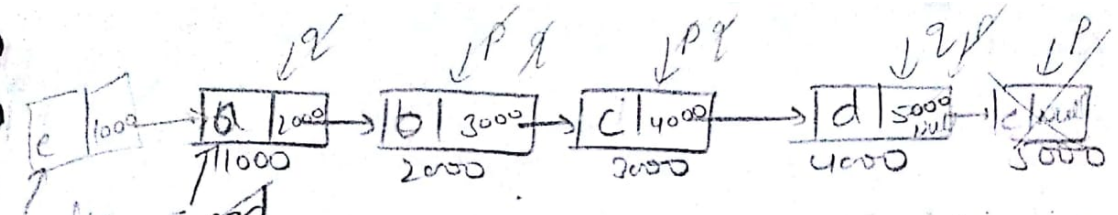
node (*) modifiedlist (node *head)
head: value | next

returns address of node type

node *p, *q;

if (head == Null) || (head -> next == Null)
return (head); (L.L. having one element last = first)

Scanned by CamScanner




```
else
{
    q=null; p=head;
    while(p->next!=null)
    {
        q=p; p=p->next;
    }
    1. q->next=Null;
    2. p->next=head;
    3. head=p;
}
return(head);
}
```

I can swap these two also (∵ they are independent)

(But this can't be swapped otherwise infinite loop will generate)

Question: Write a C program to ~~sort the given linked~~ perform Binary Search on the linked list.

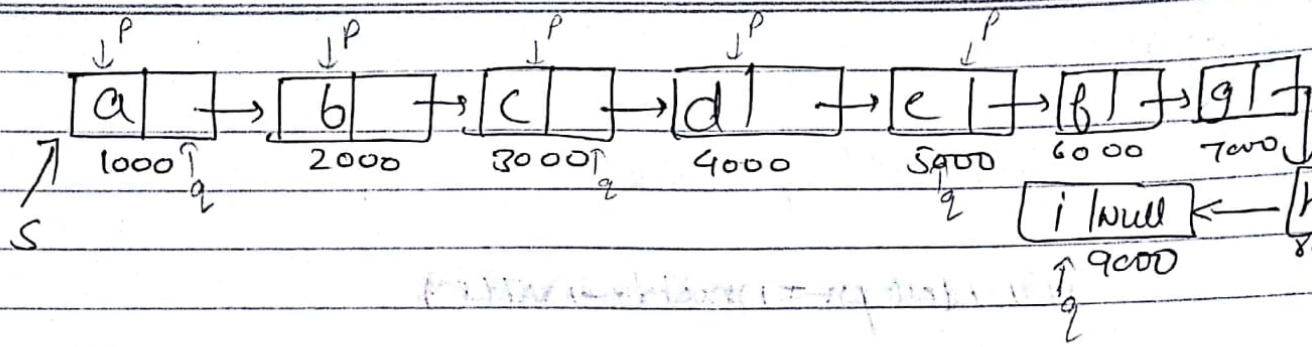
Question: Write a C program to find out middle element in the given linked list.

Interview

Scanned by CamScanner

Break Ke Baad : . . .

To get mid : p will ↑ by 1
q " p by 2 (increment fast, it will see Null first, so termination condition will be it).



algo1: p=s, c=0

```
1. while(p!=null)
   {
      c=c+1;
      p=p→next;
   }
2. c=⌈c/2⌉
3. while(c!=1)
   {
      c=c-1;
      p=p→next;
   }
```

$\Rightarrow n$ ⇒ 1 full scan to find the no. of elements in the linked list

$\Rightarrow n+1+\frac{n}{2}=\frac{3n}{2}=O(n)$

extra space = 6 Bytes

algo2:

p	q (to get mid)
1	2
2	4
3	6
4	8
⋮	
$n/2$	n

To get 1/3rd

p	q
1	3
2	6
3	9
4	12
⋮	
$n/3$	n

(only 1 scan & 2 pointers)

Scanned by CamScanner

starting address of L.L

```
Mid (s)
{
    p = s, q = s;

    ~~while (q->next->next != Null)~~
                                    "SHORTCIRCUIT"
    while ( q != Null && q->next != Null && q->next->next != Null)
            q available   after q one person available   after q 2 people available
    {
        p = p->next;
        q = q->next->next;
    }
    return (p);
}
```

$O(n) \Leftarrow n/2 \Leftarrow$

Question Write a C program to perform the "Binary Search" on the given linked list

```
BSLL( Structnode *s, x)
{
    if (s->next == Null)
    {
        if (s->data == x)
            return (s);
        else
            return (Null);
    }
    else
    {
        m = Mid (s);        // middle element address; O(n)
        if (m->data == x)   // O(1)
            return m;
```

$O(1) \Leftarrow$

Scanned by CamScanner

```
else
{
    if(m->data > x) => O(1)   (left side)
    {
        m->next = null;
        BSLL(S, x);
    }
    else
        BSLL(m->next, x);   (Right side)
}
}
```

:-)

recurrence relation:-

$$T(n) = \begin{cases} O(1) & \text{if } n=1 \\ n + O(1) + O(1) + T(n/2) = \underbrace{n + T(n/2)}_{\text{dec. G.P. series}} & \text{if } n>1 \end{cases}$$

$$= n\left[\frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + \cdots\right]$$

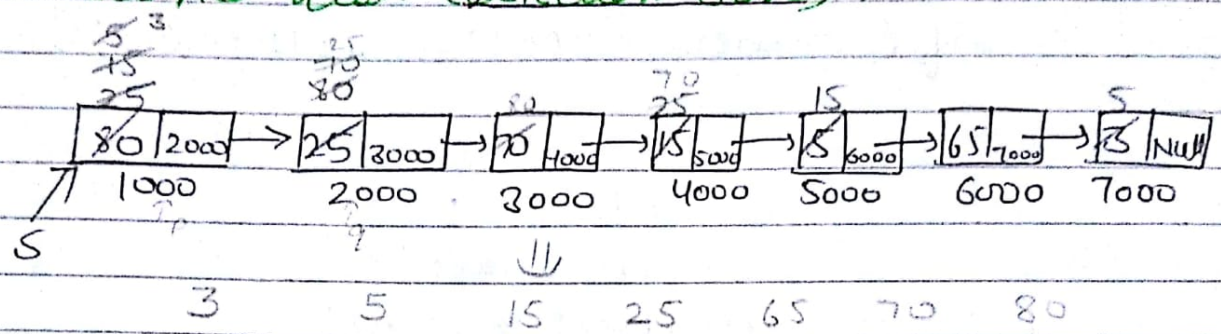
$$= O(n)$$

Note: Binary Search is possible in the linked list which will take $\Theta(n)$ (B.C. = A.C. = W.C.).

Comparing Binary Search on the arrays & linked list, linked list is not preferred.

Scanned by CamScanner

Question Write a C program to sort the given linked list. (Selection sort).



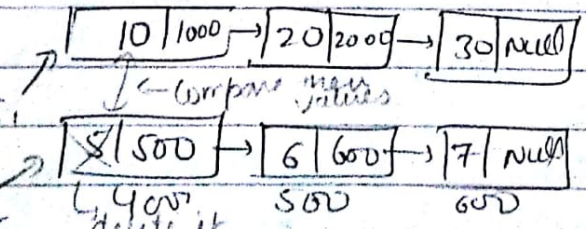

```
sort (structnode *s)
{
for(p=s; p!=Null ; p=p->next)
{
    for(q=p->next; q!=null; q=q->next)
    {
        if(p->data > q->data)
            swap(p->data, q->data);
    }
}
```

$O(n)$ ← $O(n)$

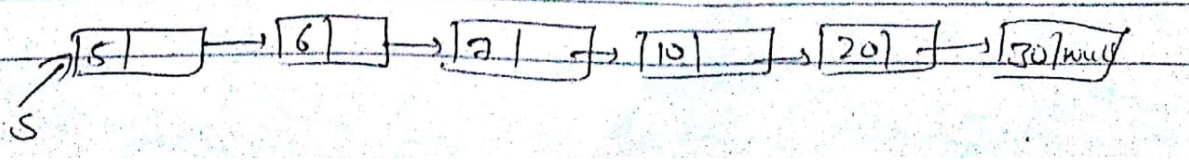
T.C. = $O(n^2)$ (Sum of n natural no.)

Extra space = 8 bytes = $O(1)$ (2 local variables)

Merge Sort (Merging two linked lists)



- Merge time = $m+n$ (numbers)
- No extra space taken (∵ just linking)



Scanned by CamScanner

Quicksort in L.L. ⇒ $\Omega(n \log n)$
$O(n^2)$

```
ms(S)
{
    if (S->next == NULL)  return(S);
    else
    {
        m = Mid(S);  ⇒ O(n)
        p = m->next;
        m->next = NULL;

        ms(S);
        ms(P);
        merge(S, P);  // Combine Step
        return(S);
    }
}
```

recurrence relation

$$T(n) = \begin{cases} O(1) & \text{if } n=1 \\ \underbrace{n}_{\text{mid finding}} + T(n/2) + T(n/2) + \underbrace{n}_{\text{Combine Time}} \end{cases}$$

$$\Downarrow$$

$$2n + 2T(n/2) \Rightarrow O(n \log n)$$

Note: Merge Sort algorithm in the linked list will take $O(n \log n)$ Time Complexity because recurrence relation is

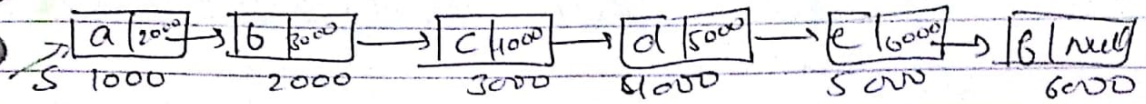
$$T(n) = 2T(n/2) + 2n \Rightarrow O(n \log n).$$

$$\Downarrow$$

[mid (divide time), merge time (Combine time)]

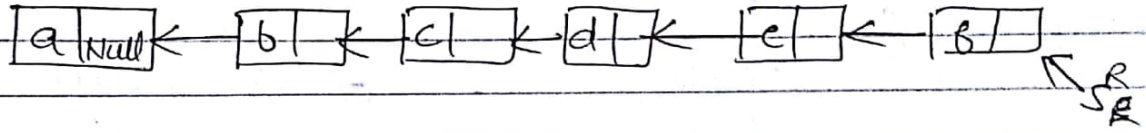
Ques: Write a C program to reverse the given linked list.

Scanned by CamScanner



- Structure node *a[10]; //array of structures
- Struct node *a[10]; //array of 10 struct node type pointers
- Struct node *a(); //function which return struct node type pointer

Reversed list (reverse the links)

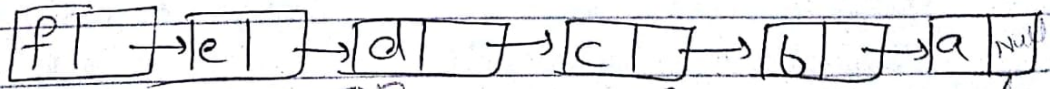


@Algorithm 1:

Take 1 node & add it in first place.

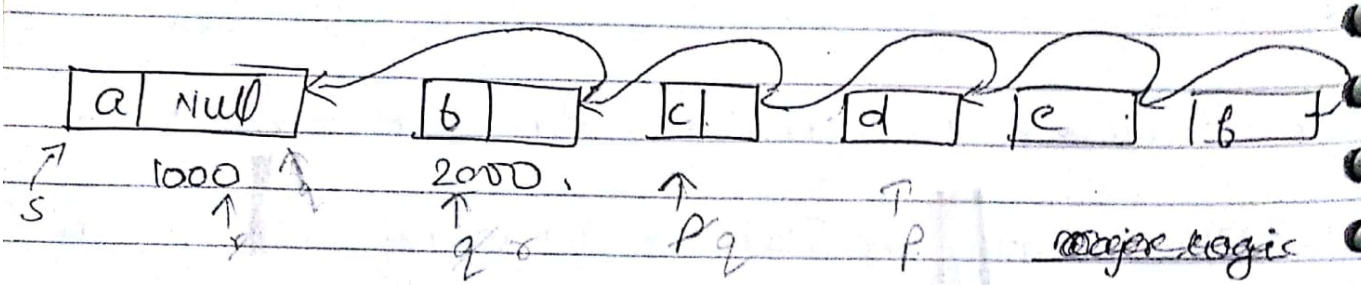
- Delete 1st node from S & add to S^R at Start. $\Rightarrow O(1)$
- Continue n times $\Rightarrow O(n)$
  $\hookrightarrow O(n)$

@Algorithm 2



- Delete last node from S and add to S^R at last. $\Rightarrow O(n^2)$
  $\hookrightarrow n - 2n$
- Continue n times

@Algorithms (Go along with your friends & problem solved).

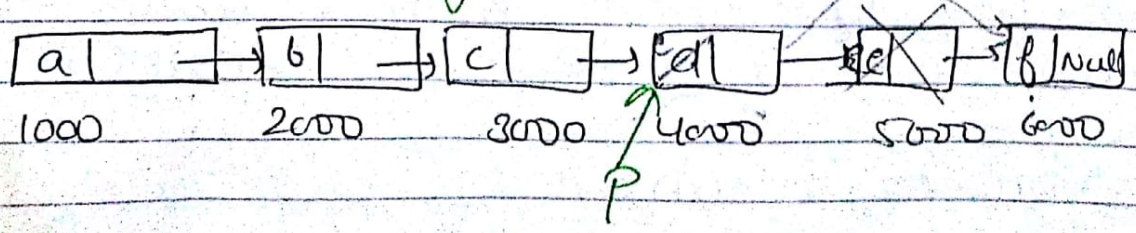


~~major logic~~

```
Reverse(S)
{
    r=q=Null; p=S;
    while (p!=Null)
    {
        r=q;
        q=p;
        p=p->next;
        q->next=r;
    }
    S=q, return(S);
}
```

⇒ O(n) time

Ques: Consider the following Linked List (only pointer P is given). From the above linked list delete a node pointed by pointer P.

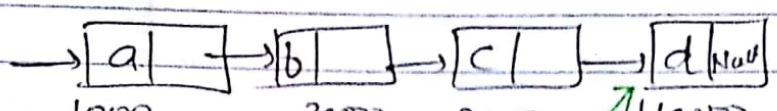


Scanned by CamScanner

→ I can't go back from p. so, I will copy the q data to p & then I will delete q.

```
q = p->next;
p->data = q->data
p->next = q->next;
free(p);
q = Null
```

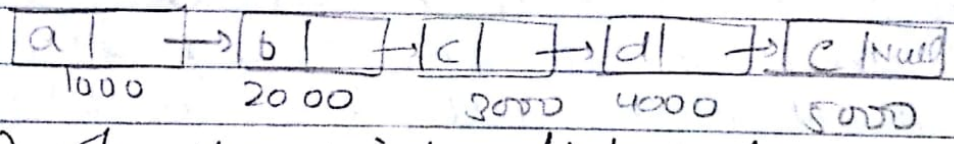
#



(Deleting the node pointing by p is not possible)

(∵ we can't go forward or backward)

Drawbacks of Single Linked List :-



1). In the single Linked List every node contain only one pointer & that pointer also pointing to next node that means we can always go only in one direction i.e. we can't go back.

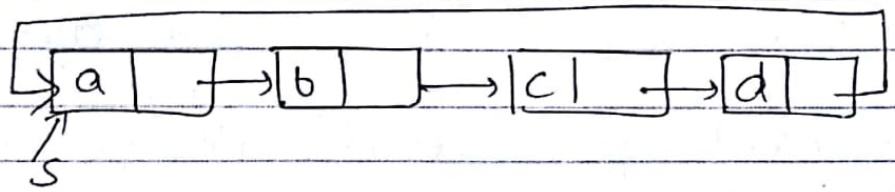
2) In Single linked list last node next part always contain null that means we are not utilizing it properly.

To eliminate the above two drawbacks we are going to the Circular Single linked list.

Scanned by CamScanner

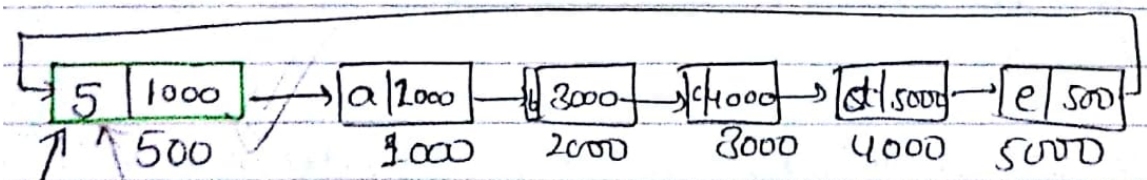
Circular Single linked list

1. In the single L.L. last node next part if we store first node address then we call it CSL.



2. The advantage with circular S.L. is we can go back also but it is costlier.
3. The biggest drawback with circular S.L.L. it may go to the ∞ loop if we don't write the code properly. (cycle)

Note: using HEAD node we can eliminate ∞ loop / cycle in the given C.S.L.L.



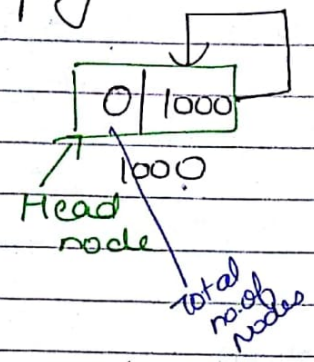
Head node (total no. of nodes in the CSLL) (Special node meant for stopping the loop)

- To count total no. of nodes in CSLL ⇒ O(1).

- Head node data part will contain valuable info of the given L.L. like total no. of nodes in the given L.L.

Scanned by CamScanner

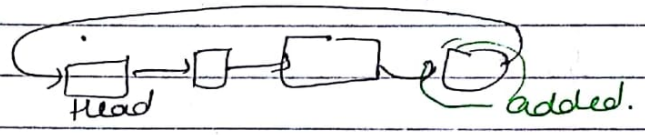
When Circular S.L.L. with head node is empty



(1000 is pointing to itself)

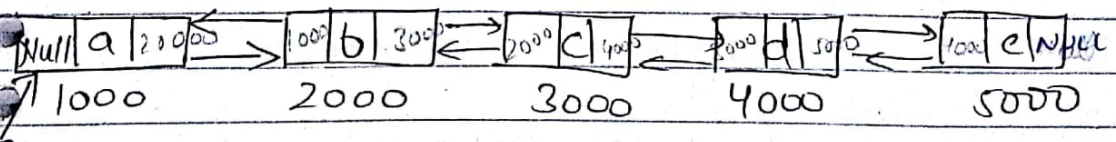
if (Head → next = Head)
then empty C.S.L.L.

To add a node in C.S.L.L. at end



Double Linked List

```
Struct DLL
{
    StructDLL *lp;      — left pointer
    int data;
    StructDLL *rp;      — right pointer
};
```

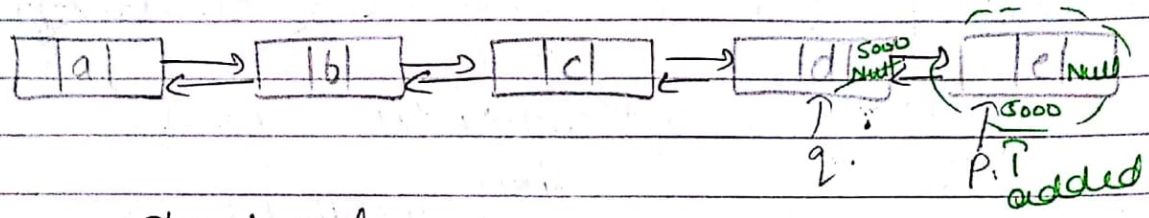


Last node: S→rp = Null
Start node: S→lp = Null

In every node it has taken 4 Bytes extra (for the 2nd pointer)

- To come from last node to first node ⇒ O(n)
- Best in terms of time

Question Write a C program to Insert a node with data node x at the end of the given double linked list.



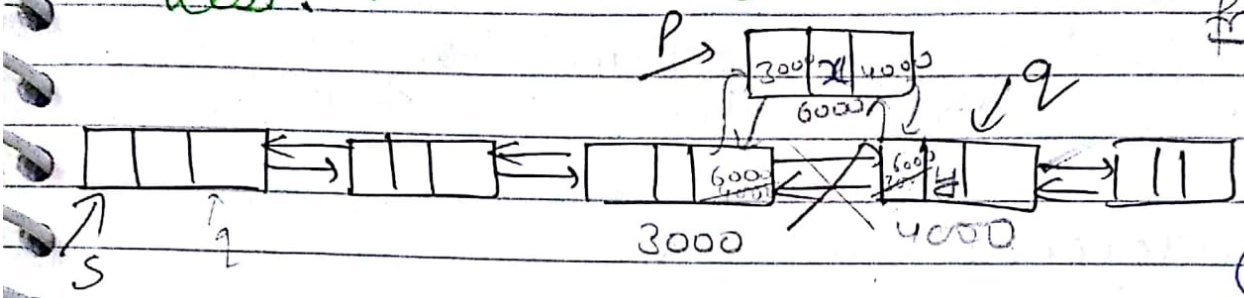

```
Struct node DLL
{
    Struct DLL *lp, *rp;
    int data;
};

Insert at end (S, x)
{
    p = (Struct DLL*) malloc (Sizeof (Struct DLL));
    p->data = x;
    p->lp = p->rp = Null;

    1. If (S == Null)
        { S = p;
          return
        }
    2. q = S;
       while (q->rp != Null)
       {
           q = q->rp;
       }
       q->rp = p;
       p->lp = q
       return(S);
}
```

Scanned by CamScanner

Question

Write a C program to Insert a node with data x before a node with data y in the given double linked list.

(No need of friend)



No need of friend

Insert x before y (s, x, y)

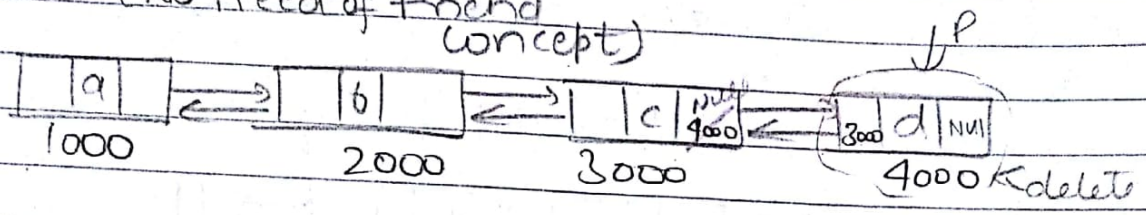
```
{
p = (struct DLL*) malloc(sizeof(struct DLL));
p->data = x;  p->lp = p->rp = NULL;
1. if (S == NULL)  { S = p;  //Checking
                     return; }  DLL is empty or not
2. q = S;
3. While (q->data != y && q->rp != NULL)
   {
     q = q->rp;
   }
   p->rp = q;  p->lp = q->lp;
   3. q->lp->rp = p;
   4. q->lp = p;
}
```

You can't swap these 2 statements.

Scanned by CamScanner

Question

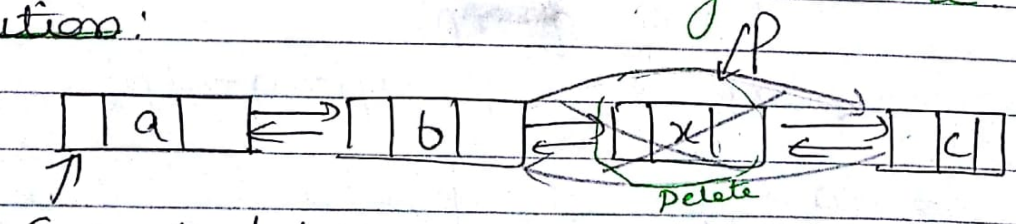
Write a C program to delete a node at the end of D-LL.
(No need of friend concept)




```
Deletelast (s)
{
    1. empty
       p=s;
    2. while(p->rp != Null)
       {
           p=p->rp;
       }

       p->lp->rp=Null;
       free(p);
       p=Null;
    return(s);
}
```

Question: WAC program to delete a node with data x in given DLL

Solution:




```
Deletex (S, x)
{   1. empty
    2. if (s->data==x)
       {
           s=s->rp;
           free(s->lp);
           s->lp=NULL;
       }
```

Scanned by CamScanner

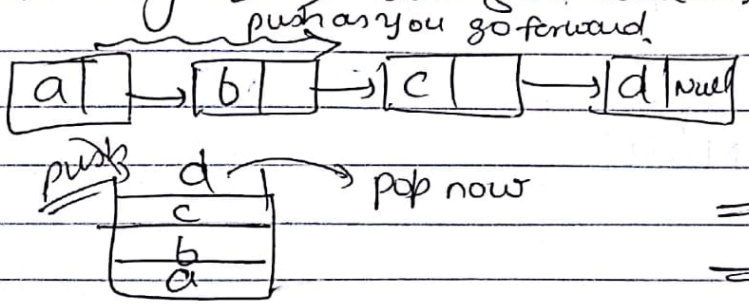
A special type of DLL ⇒ Circular D.L.L.
" " " SLL ⇒ Circular S.L.L.

Concept

The Greatest advantage of Circular D.L.L or D.L.L is :-

- If you lost one pointer with the help of another pointer we can get it back (Recovering the lost pointer)

In Single L.L., to get the reverse elements u need one extra data structure (stack)

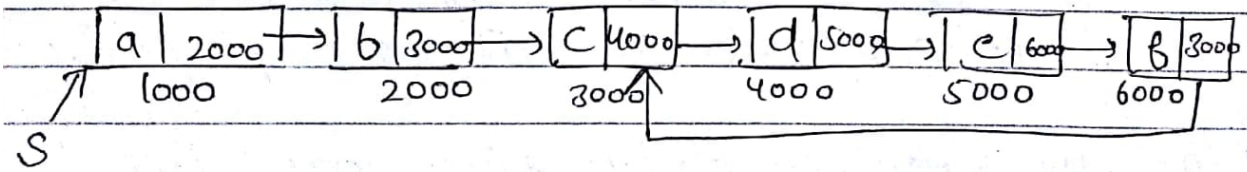


⇒ O(n) T.C.

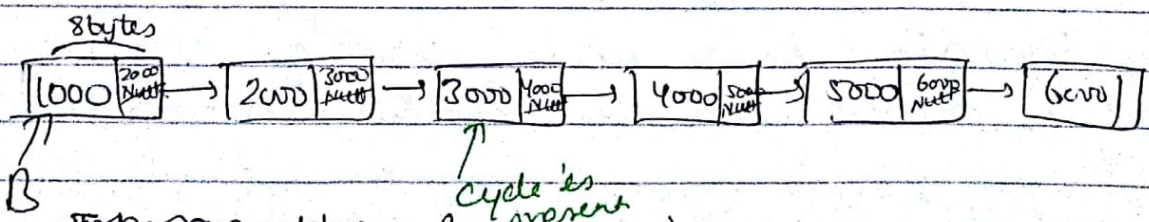
⇒ O(n) S.C.
↳ Extra space of stack = n
L.L. = n

Question

Write a C program to find ∞ loop in the given Single Linked List.



Algo 1:



~~Take one address & pick one is~~

T.C. = O(n²)

calloc: Create m/m, by default zero is placed in it.
(for flag use calloc m/m allocation) 140

malloc: create m/m, by default Garbage is initialized to it.

Algo 1:

→ Take every node address of the given L.L. & add to the visited linked list (B) if that Node address is not visited. ⇒ n

↳ checking purpose (use Linear Search) ↳ O(n)

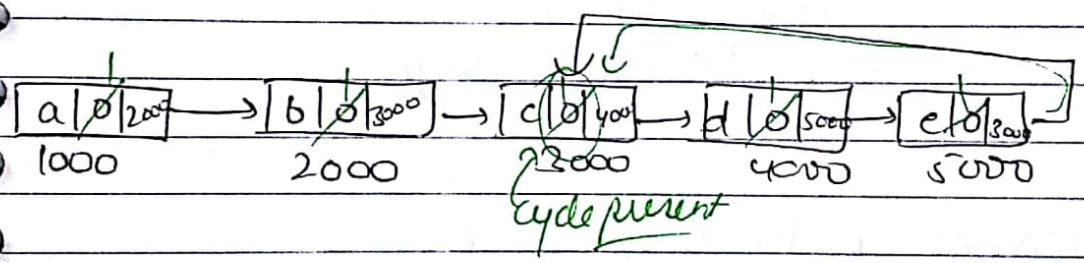
else

print that node.

T.C. $O(n^2)$

space comp = 8n = O(n)

Algo 2: use flag in the data structure of the linked list if modification is allowed.



Note: Use flag counter whenever you have to find the CYCLE!

Modify the every node of the Linked List by adding an extra field called flag which is initally set to zero. ⇒ O(n)

→ Goto every node & Check every node's flage is zero/one.

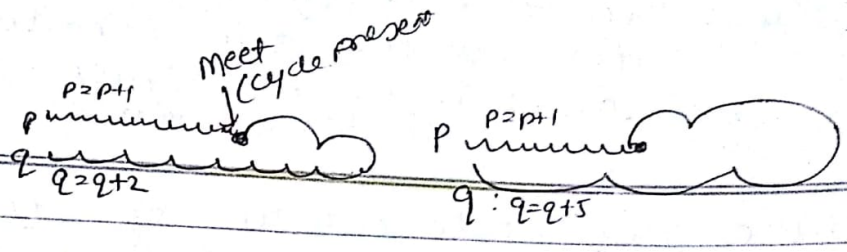
If (flage=0) make flag=1

else loop is found. ⇒ O(n)

S.C. ⇒ O(n) ← n flag (each of 1 byte)

O(n) T.C.

Scanned by CamScanner



Algo 3:

1. Take two pointers A & B.
   - A will increment by 1
   - B " " by 2.
2. If (A == B)
   loop found
   else
   goto step 2.

O(n) Time

space O(1)
(2 pointers only)

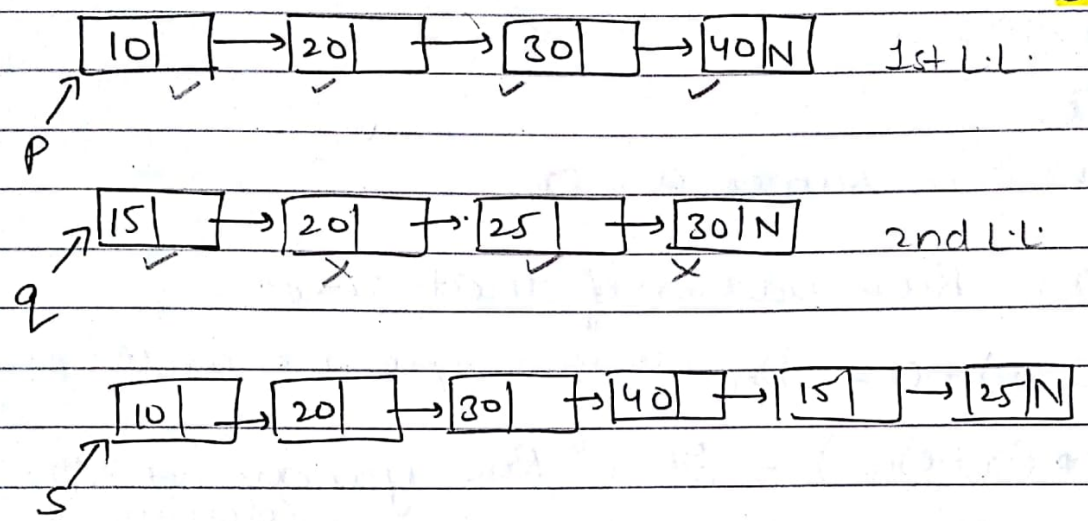
Scanned by CamScanner

$*a+1 = 1002$

Row selection over

1 element of that row is skipped.

Question Write a C program to perform the union of two linked list. (slowest)



Algo:

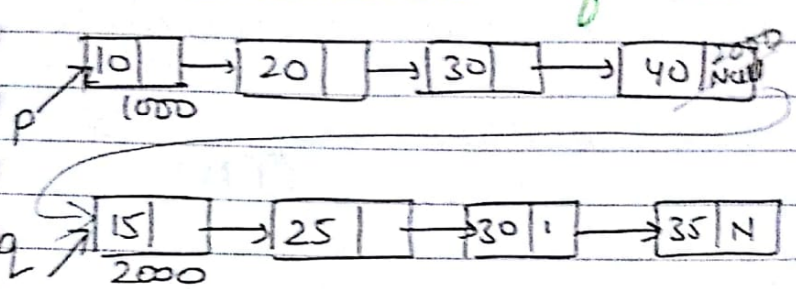
1. Add first linked list to the result (n elements added to the resultant list) $O(n)$

2. Take every element of the 2nd L.L. and check it is available in the first L.L. or not using linear Search. If, it is available then don't add else add. $O(n^2)$

$O(n^2)$

Scanned by CamScanner

Question Write a C program to perform concatenation of two linked list. (Fast)



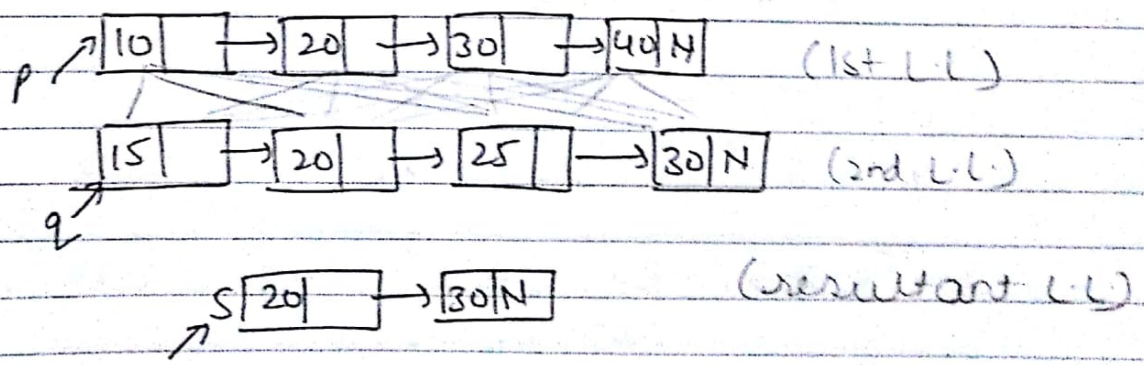
Algorithm :-

① Goto the last node of the first L.L. $O(n)$

② Link the last node with the first node of the 2nd L.L. $O(1)$

$O(n)$

Question Write a C program to perform the Intersection of two linked list. (Slowest)



Algorithm

1. Every element of 1st L.L is compared with every element of the 2nd L.L. If match found then add to the result else goto next ele

$O(n^2)$

Scanned by CamScanner

Membership of L.L.

Is x is an element of the given L.L.?
(simply checking all the elements of L.L.)
$O(n)$

Polynomial Addition

Ex:

$$p: 10x^3 + 5x^2 + 2x + 5$$
$$q: 20x^3 + 15x^2 + 5x + 20$$
$$\overline{30x^3 + 20x^2 + 7x + 25}$$

Note: If powers are equal then only add their coefficients otherwise don't add.

Ex

$$p: 10x^{15} + 20x^{35} + 3x^2 + 6$$
$$q: 25x^5 + 77x + 10$$

Highest power ele is written 1st.

$$\overline{10x^{15} + 45x^5 + 3x^2 + 77x + 16}$$

Ex

$$p: 10x^{15} + 20x^{12} + 3x^{10} + 5x^6$$
$$q: 25x^8 + 77x^6 + 10x^3 + 20x^2 + 100$$

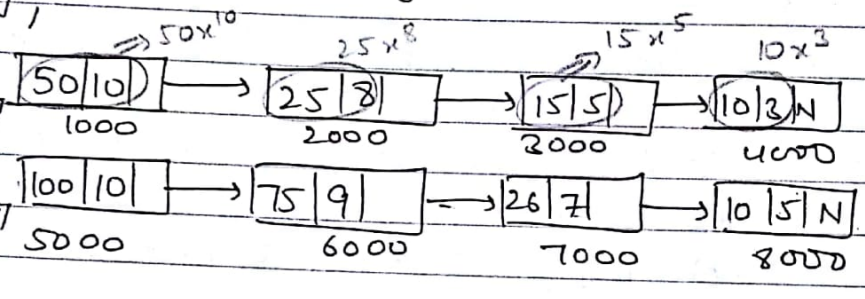
copy rest.

$$\overline{10x^{15} + 20x^{12} + 3x^{10} + 25x^8 + 82x^6 + 10x^3 + 20x^2 + 100}$$

Note: Continue the while loop until only one L.L. is over. If any one L.L. is over then copy the 2nd L.L. rest elements.

Scanned by CamScanner

```
struct poly
{
    int coe;
    int pow;
    struct poly*next;
};
```




```
while (p!=Null && q!=Null)
{
    if (p->pow == q->pow)
    {
        R->pow = p->pow;
        R->coe = p->coe + q->coe;
        p= p->next;
        q= q->next;
        R = R->next;
    }
    else
    { if (p->pow > q->pow)
      {
        R->pow = p->pow;
        R->coe = p->coe;
        R = R->next;
        p= p->next;
      }
```

Resultant

Scanned by CamScanner

```
else
{
        R->pow = q->pow;
        R->coe = q->coe;
        R = R->next;
        q = q->next;
}
}
}
```

// At the end of while loop I will ask whether q is over or p is over. If anyone is over then copy the next one data.

```
if (p==Null && q==NULL)
        return(R);

else
{
        if (p==NULL)
        {
                add q to R and return R;
        }
        else
        {
                add p to R and return R;
        }
}
```

Time Complexity of polynomial addition

Worst case: $O(m+n)$ (1st & 2nd L.L are simultaneously over)

Best case: $O(n)$ (only one L.L is over & Rest are copied)

Scanned by CamScanner

Polynomial Multiplication

Logic: multiply coefficients & add power

Ex. $\underset{p}{20x^5} \times \underset{q}{10x^6} = \underset{r}{200x^{11}}$

for every term of p u have to multiply with every term of q.

```
while (p!=Null)
{
    while (q!=Null)
    {
        R->pow= p->pow + q->pow;
        R->coe= p->coe * q->coe;
        R= R->next;
        q= q->next;
    }

    p= p->next;
}
```

T.C: $O(n^2)$ (2 loops are present

Calloc (size (struct node))

data	next
0	0

(calloc() func^n. by default stores '0' in the node

malloc (size(struct node))

data	next
g	g

(malloc() func^n. by default stores 'garbage' in the node)

Scanned by CamScanner

Question: Write a C program to implement STACK using Linked List

x_7	6
x_6	5
x_5	4
x_4	3
x_3	2
x_2	1
x_1	0

S

N=7

Initially Top = -1 (No element in stack)

(Stack using array)

Top = ~~-1~~ ~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ 6

Push

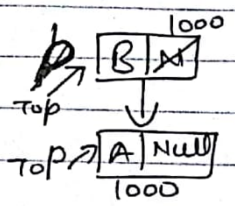
```
push (x)
  {   if (Top==N-1)
            stop; //stack full
      else
        {   Top++;
            S[Top]= x;
        }
  }
```

Pop

```
pop ()
  {
      if (Top==-1)
            stop; //stack empty
      else
        {   x= S[Top];
            Top--;
            return (x);
        }
  }
```

Scanned by CamScanner

- Deletion order in stack is LIFO.

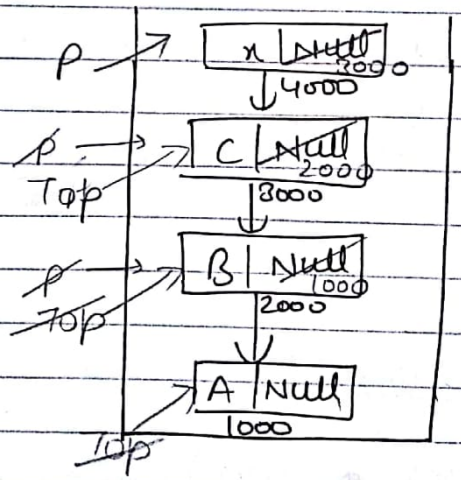


p→next = Top;
Top = p;

(B should be linked to A ∵ it will be deleted first during pop)

dynamic stack

- only one element in the stack
  Top→next = Null.
- No element in stack
  Top = Null.



Push

(Stack using linked list)

```
Struct Stack
  {
      int data;
      Struct stack * next;
  };
```

c

```
push (structstack *Top, int x)
{
        Structstack *p;

    p=(structstack*) malloc (sizeof (structstack));
                                    create m/m

        if (p==NULL)                    m/m is not available
        {
            Printf ("m/m is over");
            exit(1);
        }
    // if p≠Null then somewhere m/m is created

        p->data = x;
        p->next = Top;
        Top=p;   // Top is changed

}   . You can Insert the ele with m/m is available
```

Pop (use friend concept)

```
Pop( Struct stack * Top)
{
        Struct stack *p; //my friend

        if(Top==NULL)
        {   printf(" Stack is Empty");
        }
    //Top ≠ Null
        p=Top;
        Top= Top->next;
        free(p);
        P=Null;
}
```

Scanned by CamScanner

Question Write a C program to Implement Queue (FIFO) using Linked List.

a	b	c	d	e	f	g	h	i	j
0	1	2	3	4	5	6	7	8	9

N=10

f = 0 ~~1~~ .... (pos. of element to be deleted)

R = 5 ~~6~~ ~~7~~ ~~8~~ ~~9~~ (pos. of newly Inserted element)

Insert

```
if (R==N-1)
        stop; // Queue is full

else
{ if(f==R==-1)
    {
        f=R=0;
    }
  else
    {
        R++;
    }
  Q[R]=x;
}
```

until now no element in Q we are doing 1st Insertion so make the front & Rear to come inside the Queue.

Delete

```
if (f==-1)
        stop; // Q is Empty

else{
```

```
if(F==R)   // Q has 1 element
  {  x = Q[F];
     F = R = -1;
  }
else
  {  x = Q[F];
     F++;
  }
}
```

Q

a	b	c	d	e
0	1	2	3	4

N = 5

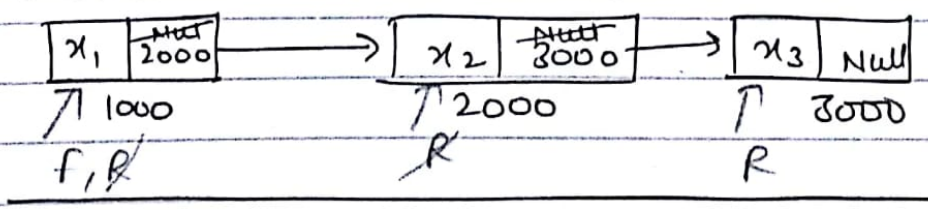
F = ~~-1~~ 0 ~~1~~ ~~2~~ 3 -1

R = ~~-1~~ ~~0~~ ~~1~~ ~~2~~ 3 -1

Insertion Deletion

using Linked list applying Queue

FIFO



- F = R = Null (No element in Q)
- If Queue is given to us that means front & Rear are given to us.

Insert

```
Insert(StructQueue *F, StructQueue *R, int x)
{
    StructQueue * p;
```

Scanned by CamScanner

```
p=(struct Queue *) malloc (sizeof (struct Queue));
                    //somewhere for p m/m is created.

    if (p==NULL)
        {  printf ("m/m is over");
           exit(1)  //abnormal termination
        }

    p->data=x;   p->Next=NULL;
else
{ if (f==R==NULL)     //1st Insertion
                      (Both f & R have come
    {                 inside Queue, 1st Insertion
        f=R=p;        is done;
    }
  else
    {
        R->next=p;
        R=p;
    }
}
}
```

Delete

```
Delete (struct Queue *f, struct Queue *R)
{
    if (f==NULL)        (f=Null means both
                         f & R=Null)
    {
        printf("Q is empty");
            exit(1);
    }
```

Scanned by CamScanner

```
else
{
    if (f==R)   //only 1 el in Q
    {
        free(f);
        f=NULL;
    }
    else
    {
        p=f;
        f=f->next;
        free(p);
        p=NULL;
    }
}
}
```

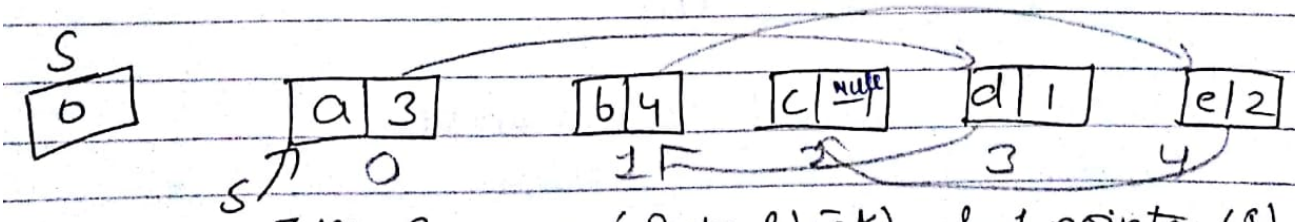
Question Write a C program to implement Linked List using arrays

Data

0	a
1	b
2	c
3	d
4	e

Link

0	3
1	4
2	-1
3	1
4	2



I/P: 2 array (Data & Link) & 1 pointer (S)

Scanned by CamScanner

Write a C program to display data part of linked list using array representation

```
display (data, link, s)
{
    int temp=s;      (s is pointing to start location)

    while (temp != -1)
    {
        printf ("%d", data[temp]);

        temp= link [temp];
    }
}
```

O/P: a d b e c

note: you can't insert or delete elements in the arrays. So deletion & Insertion are Impossible (you can't increase or decrease the array size). What only we can do is to skip the links.

Write a C program to delete last node from linked list using arrays.

```
deletelast( data, link, s)
{
    if (link [s] == -1) // L.L. has 1 element
    {
        s= -1;
        exit(1);
    }
```

(delete that element also & make L.L. empty)

Scanned by CamScanner

```
else
{
    p = -1; //my friend
    q = s;

    while (link[q] != -1)
    {
        p = q;
        q = link[q];
    }

    link[p] = -1; // After my friend no node is there.
}

}
```

Scanned by CamScanner