

CS & IT ENGINEERING

Data Structure

Linked List- 02

DPP 01

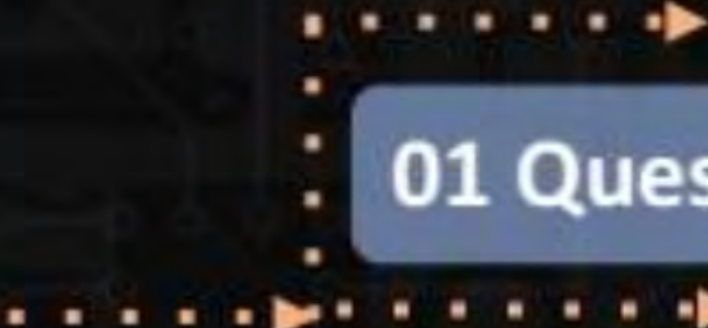
Discussion Notes



By- Pankaj Sharma sir



TOPICS TO BE COVERED



01 Question

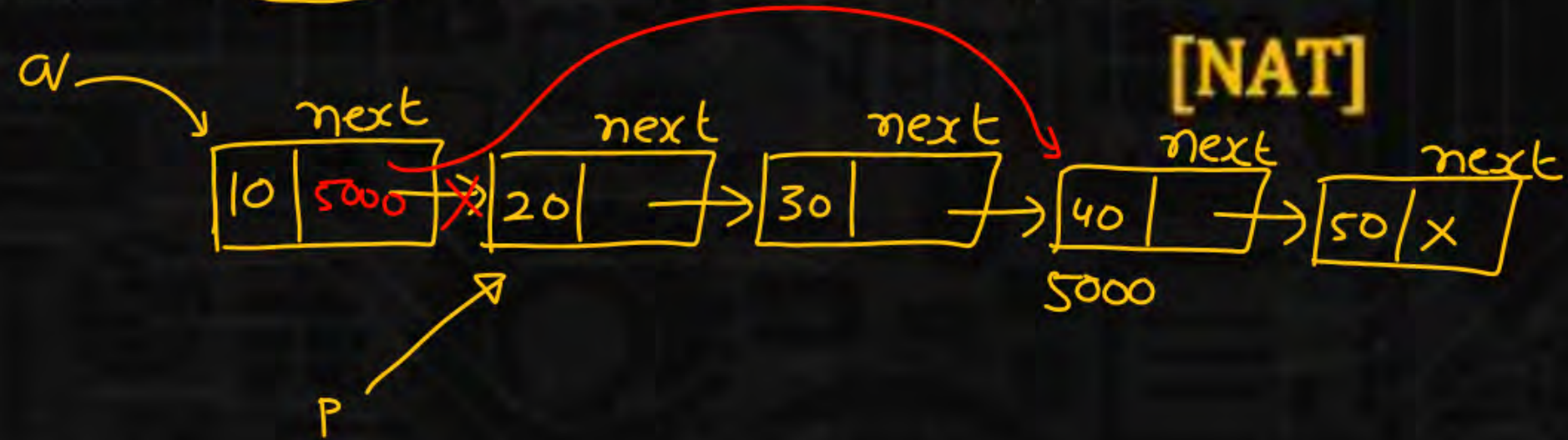
02 Discussion

Q.1



Consider a single linked list q with 2023 elements is passed to the following function:

```
struct node {  
    int data;  
    struct node *next;  
};  
void f(struct node *q){  
    struct node *p;  
    p=q->next;  
    q->next=p->next->next;  
}
```



The size of the linked list q after the execution of the function is

2023-2 = 2021

Q.2

Consider a single linked list $q['A', 'B', 'C', 'D', 'E', 'F']$ is passed to the following function:

[MCQ]

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

```
void f(struct node *q)  
{
```

```
    struct node *p;
```

```
    p = q->next->next->next;
```

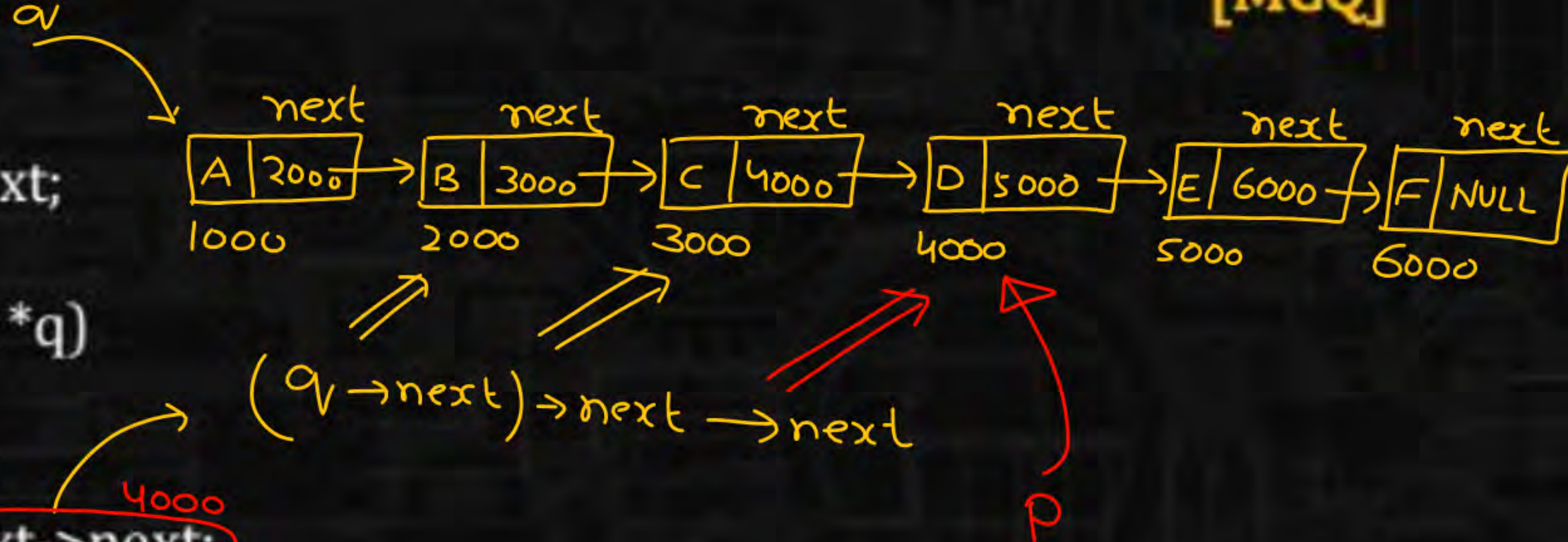
```
    q->next->next->next = p->next->next;
```

```
    p->next->next = q->next;
```

```
    printf("%c", p->next->next->next->data);
```

```
}
```

The output is-

**A.**

C

B.

D

C.

E

D.

B

Q.2



Consider a single linked list $q['A', 'B', 'C', 'D', 'E', 'F']$ is passed to the following function:

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

```
void f(struct node *q)
{
```

```
struct node *p;
```

```
p=q->next->next->next;
```

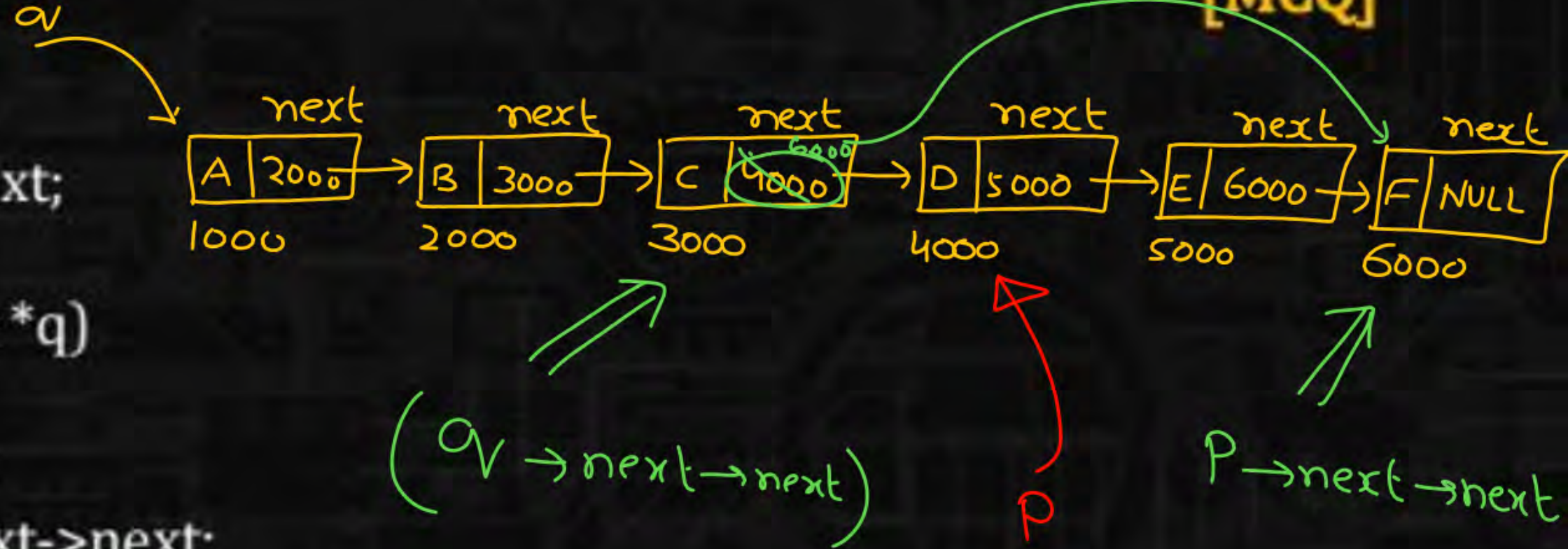
```
q->next->next->next=p->next->next;
```

```
p->next->next=q->next;
```

```
printf("%c", p->next->next->next->data);
```

}

The output is-



A.

C

B.

D

C.

E

D.

B



Consider a single linked list $q['A', 'B', 'C', 'D', 'E', 'F']$ is passed to the following function:

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

```
void f(struct node *q)
{
```

```
struct node *p;
```

```
p=q->next->next->next;
```

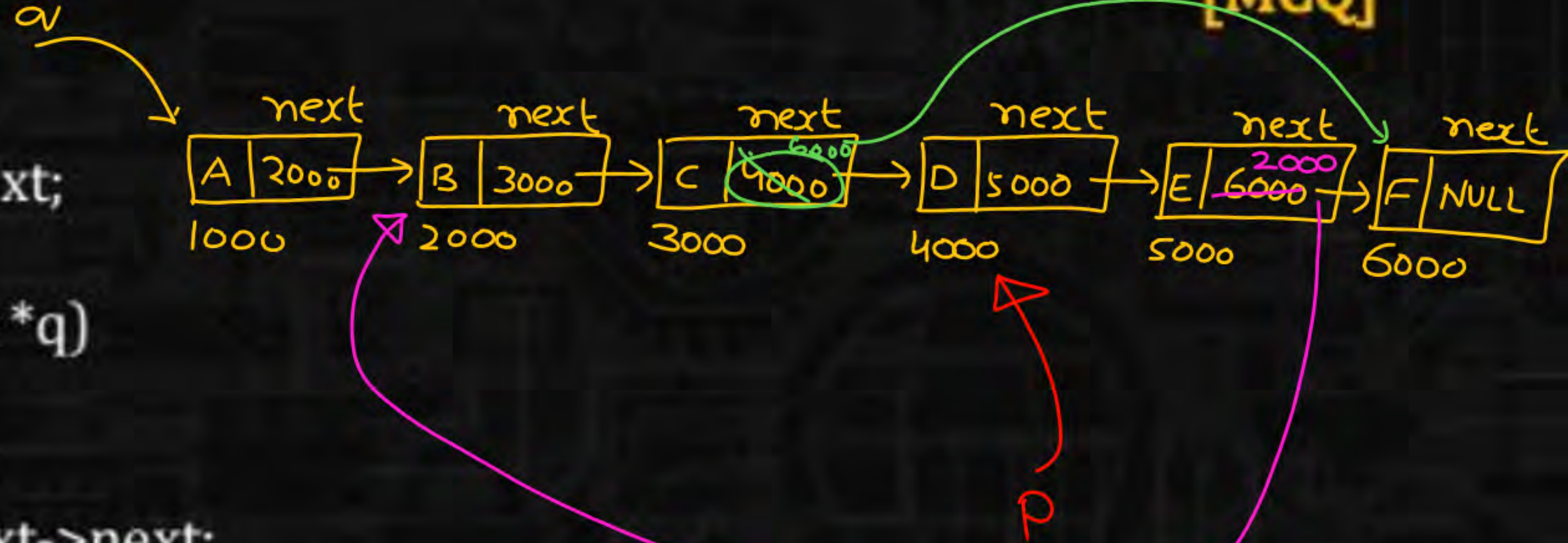
```
q->next->next->next=p->next->next;
```

```
p->next->next=q->next; (2000)
```

```
printf("%c", p->next->next->next->data);
```

}

The output is-



[MCQ]

(A)

C

A.

C

B.

D

C.

E

D.

B

Q.3



[MCQ]

Consider the following statements:

P: Linked Lists supports linear accessing of elements

Correct

Q: Linked Lists supports random accessing of elements.

Incorrect

Which of the following statements is/are INCORRECT?

A.

P only

B.

Q only

C.

Both P and Q

D.

Neither P nor Q

(B)



Q.4



Consider a single linked list $q['A', 'B', 'C', 'D']$ is passed to the following function:

[MCQ]

```
void f(struct node *q)
```

```
{
```

```
    if(q==NULL) return;
```

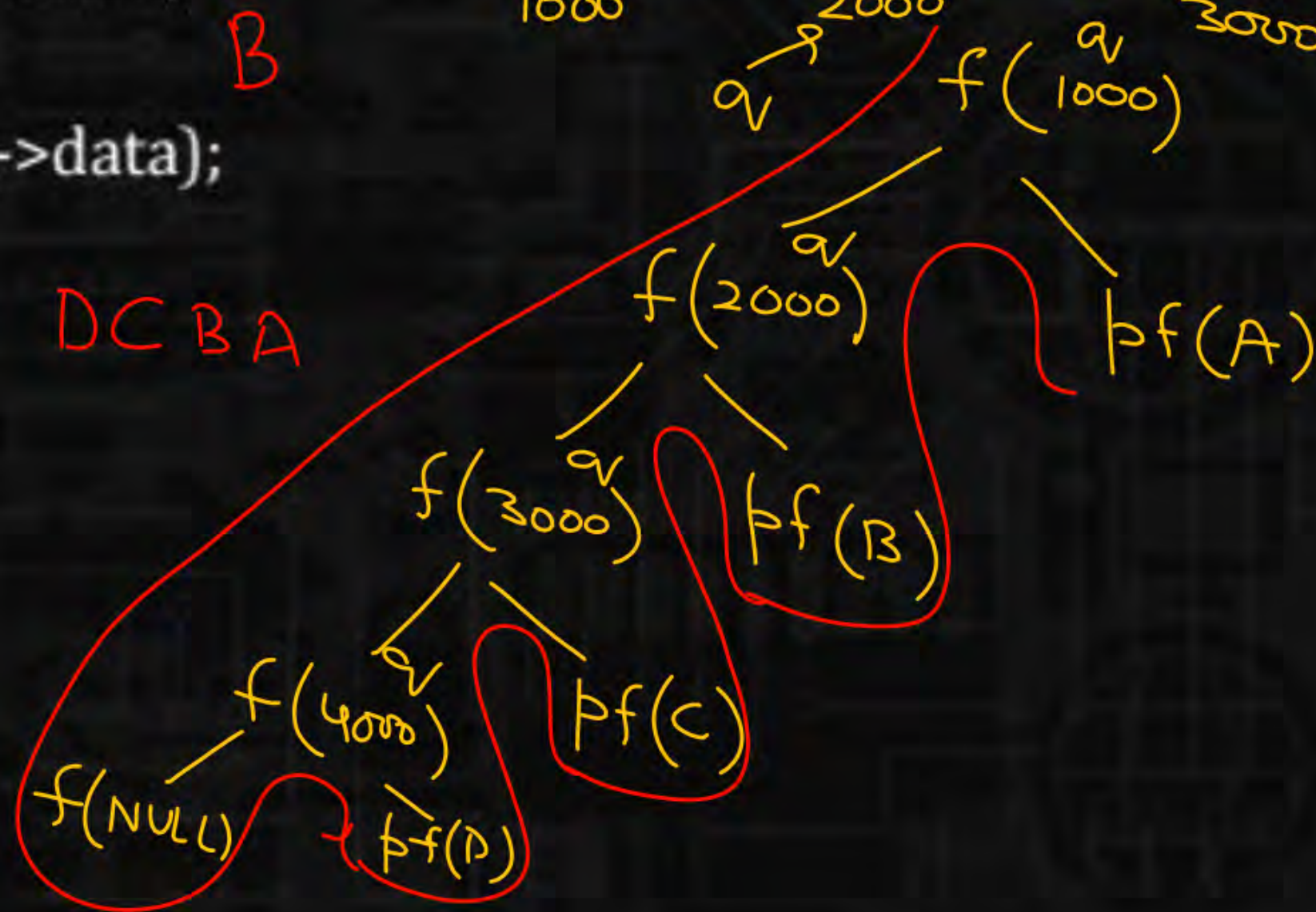
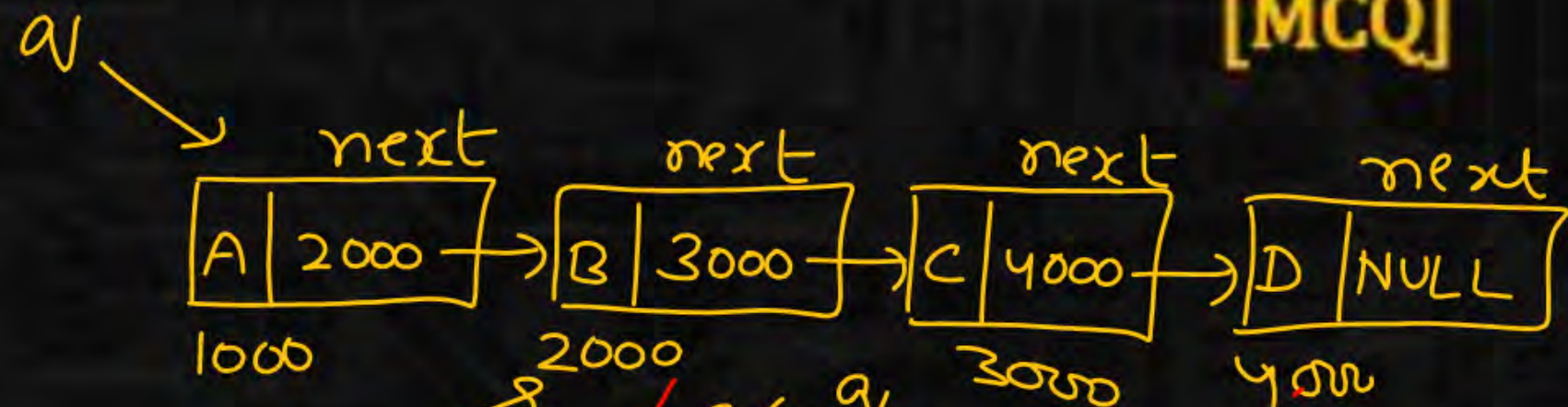
```
    1. f(q->next);
```

```
    2. printf("%c ", q->data);
```

```
}
```

The output is-

DCBA



A.

C D B A

☒ B.

D C B A

C.

A B C D

D.

B C D A

Q.5

Consider the following statements:

[MCQ]

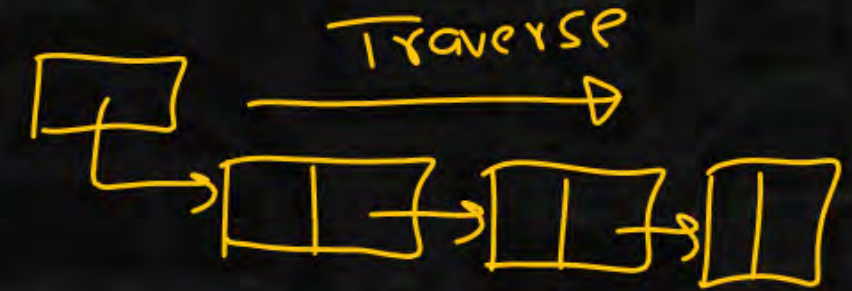


Correct

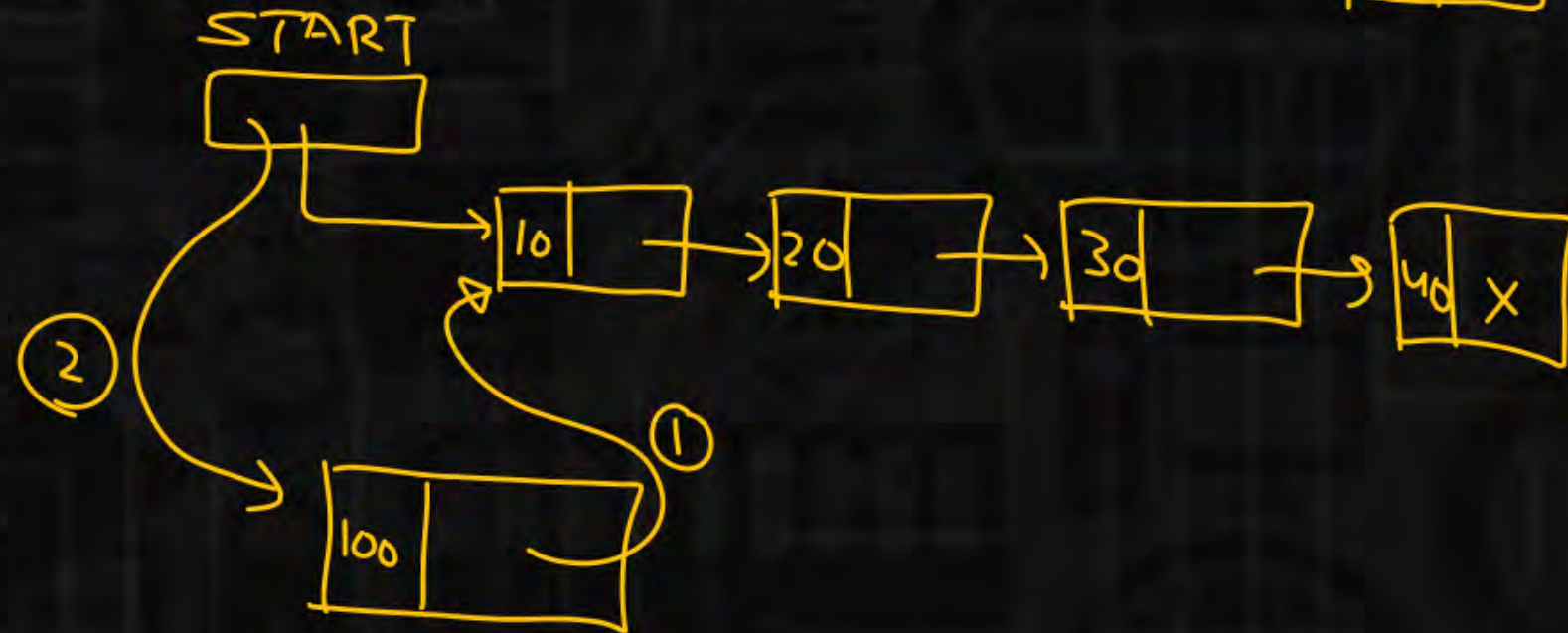
P: Insertion at the end of the linked list is difficult than insertion at the beginning of the linked list.

Q: Deletion at the beginning of linked list is easier as compared to deletion at the end of the linked list.

Which of the following statements is/are CORRECT?



- A. Both P and Q
- B. P only
- C. Q only
- D. Neither P nor Q



Q.5

Consider the following statements:

[MCQ]

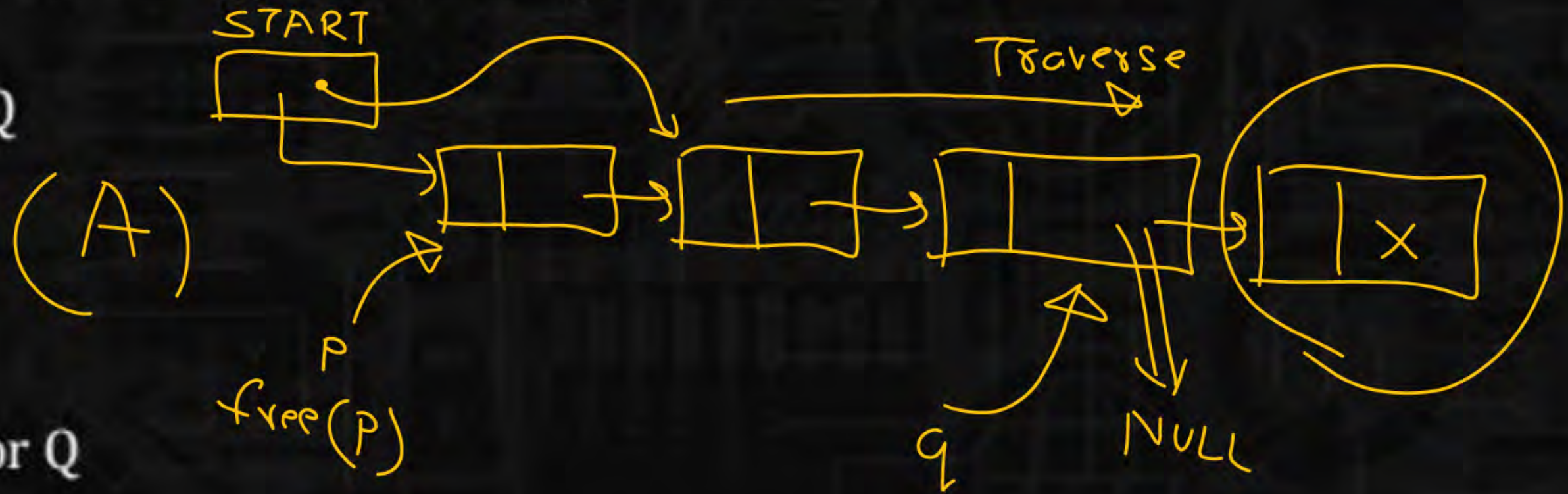


Correct P: Insertion at the end of the linked list is difficult than insertion at the beginning of the linked list.

Correct Q: Deletion at the beginning of linked list is easier as compared to deletion at the end of the linked list.

Which of the following statements is/are CORRECT?

- ☒ A. Both P and Q
- ☐ B. P only
- ☐ C. Q only
- ☐ D. Neither P nor Q



Q.6

The following C function takes a single-linked list p of integers as a parameter. It deletes the last element of the single linked list. Fill in the blank space in the code:

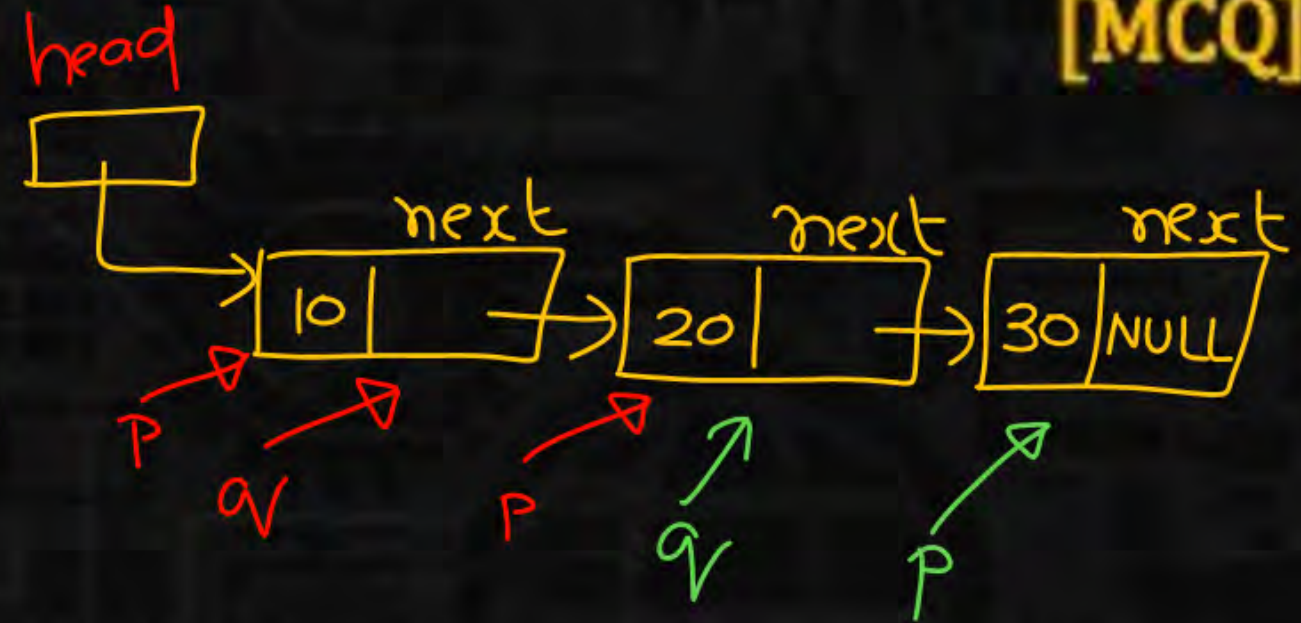
```
struct node {  
    int data;  
    struct node *next;  
};
```

```
void delete_last(struct node *head)  
{
```

```
    struct node *p=head, *q;  
    if(!head) return;  
    if(head->next==NULL){free(head);head=NULL;  
    return;}  
    while(____a____){
```

```
        q = p;  
        p=p->next;  
    }  
    ____b____;
```

```
    free(p);  
    q=NULL; p=NULL;  
}
```

[MCQ]~~A.~~

a: !head ; b: q->next = NULL; free(p)

~~B.~~

a: p->next != head ; b: q->next = q

☒ C.

a: p->next != NULL ; b: q->next = NULL

~~D.~~

a: head->next != p ; b: q->next = p

q -> next = NULL
free(p)

Q.7



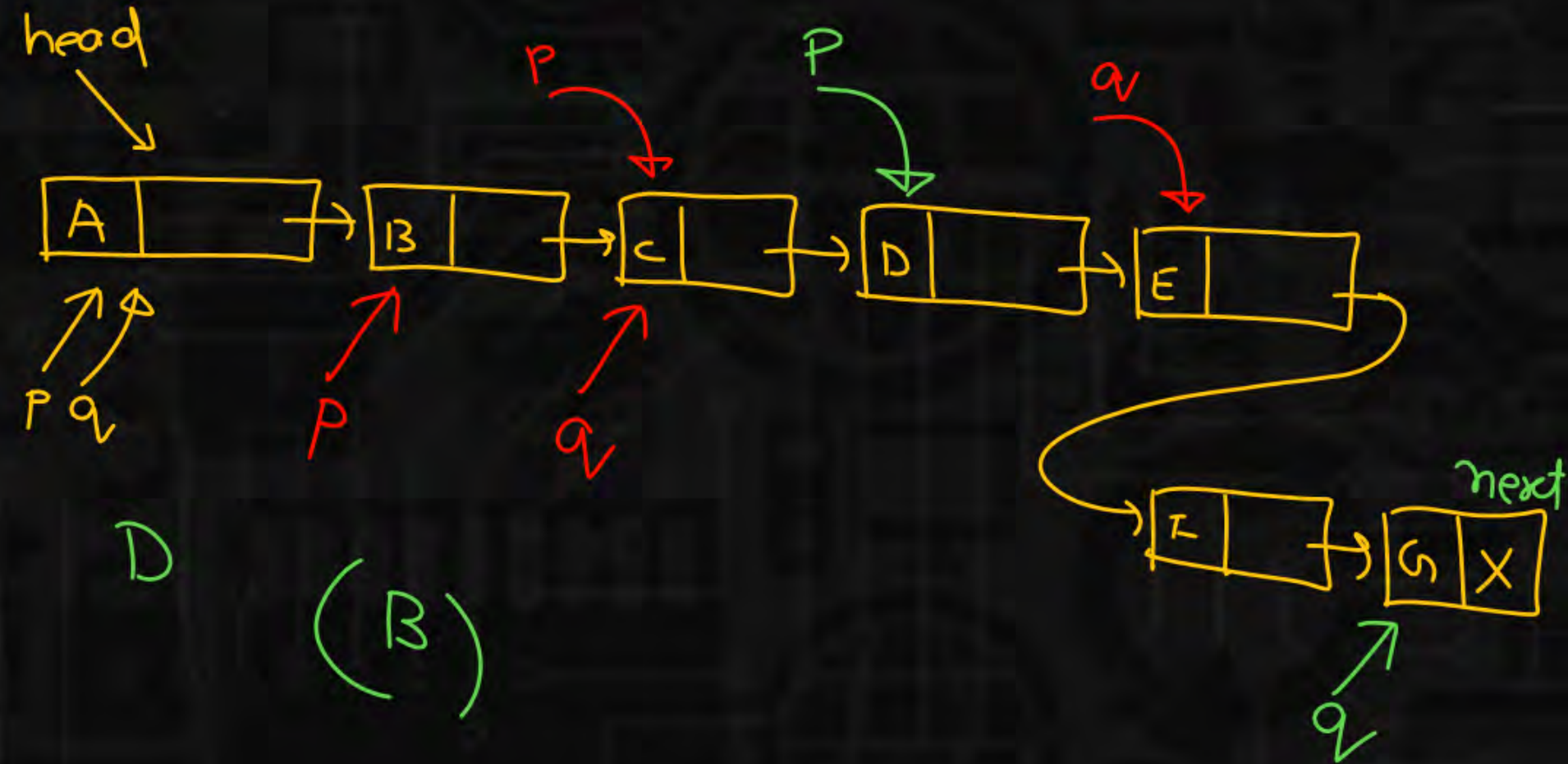
Consider a single linked list $q[['A', 'B', 'C', 'D', 'E', 'F', 'G']]$ is passed to the following function:

[MCQ]

```
void func(struct node *head){  
    struct node *p=head, *q=head;  
    while( $q \neq \text{NULL}$  &&  $q \rightarrow \text{next} \neq \text{NULL}$  &&  $q \rightarrow \text{next} \rightarrow \text{next} \neq \text{NULL}$ ){  
        p=p->next;  
        q=q->next->next;  
    }  
    printf("%c", p->data);  
}
```

The output is-

- A. C ☒ B. D
- C. E ☐ D. B



Q.8

The following C function takes a single-linked list p of integers as a parameter. It inserts the element at the end of the single linked list. Fill in the blank space in the code:

```
struct node
```

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

```
void insert_last(struct node *head, struct node *q){  
    struct node *p=head;
```

```
    if(!head) return;
```

```
    while(____a____)
```

```
        p=p->next;
```

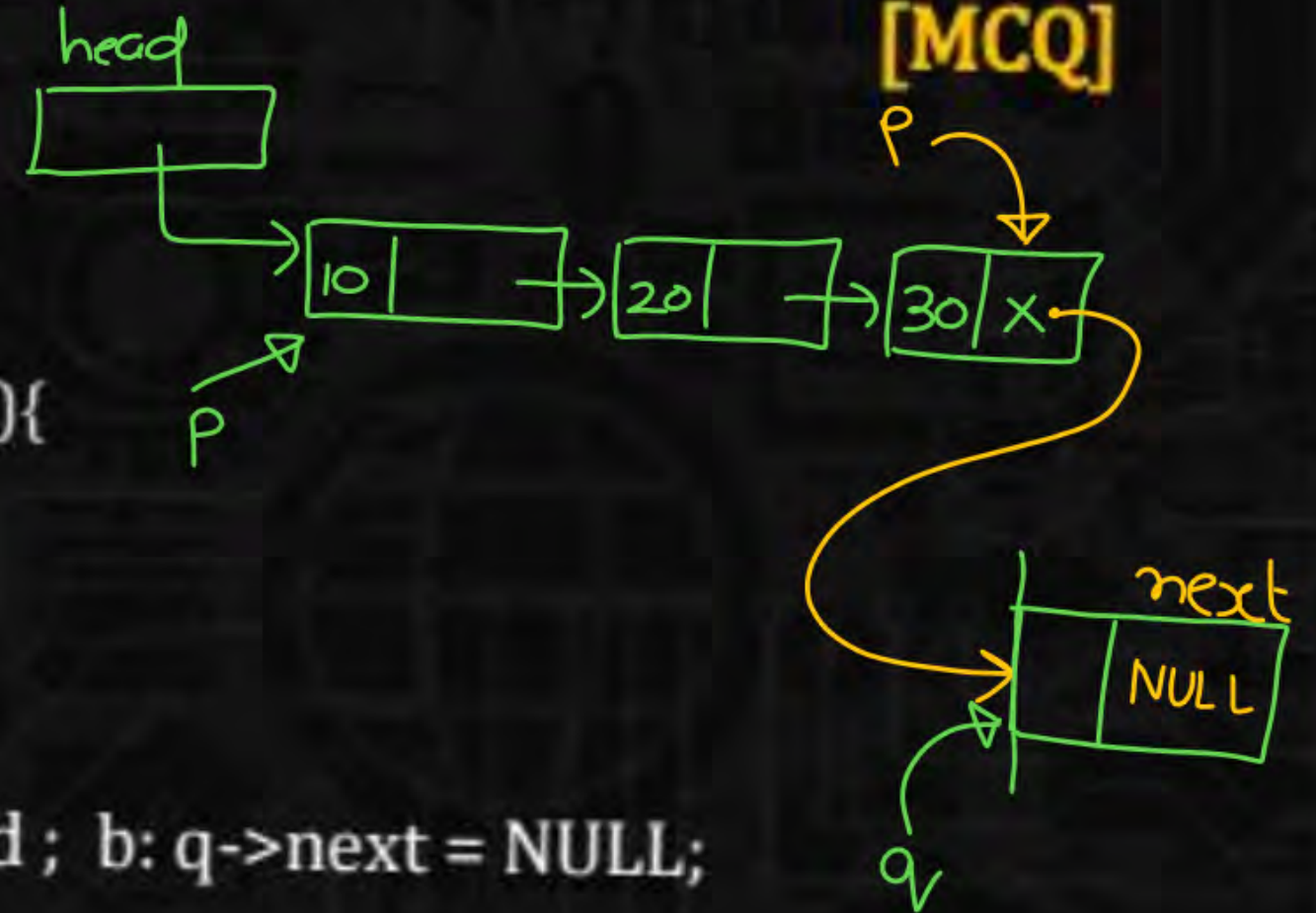
```
        ____b____;
```

```
        q=NULL;
```

```
        p=NULL;
```

```
}
```

Assume, q is the address
of the new node to be added.



~~A.~~

a: !head ; b: q->next = NULL;

~~B.~~

a: q->next != NULL; b: p->next = q

C.

a: p->next != NULL ; b: p->next = q

~~D.~~

a: head->next != p ; b: q->next = p

