CS & IT ENGINEERING

Data Structure & Programming

Linked List-1

DPP.- 01

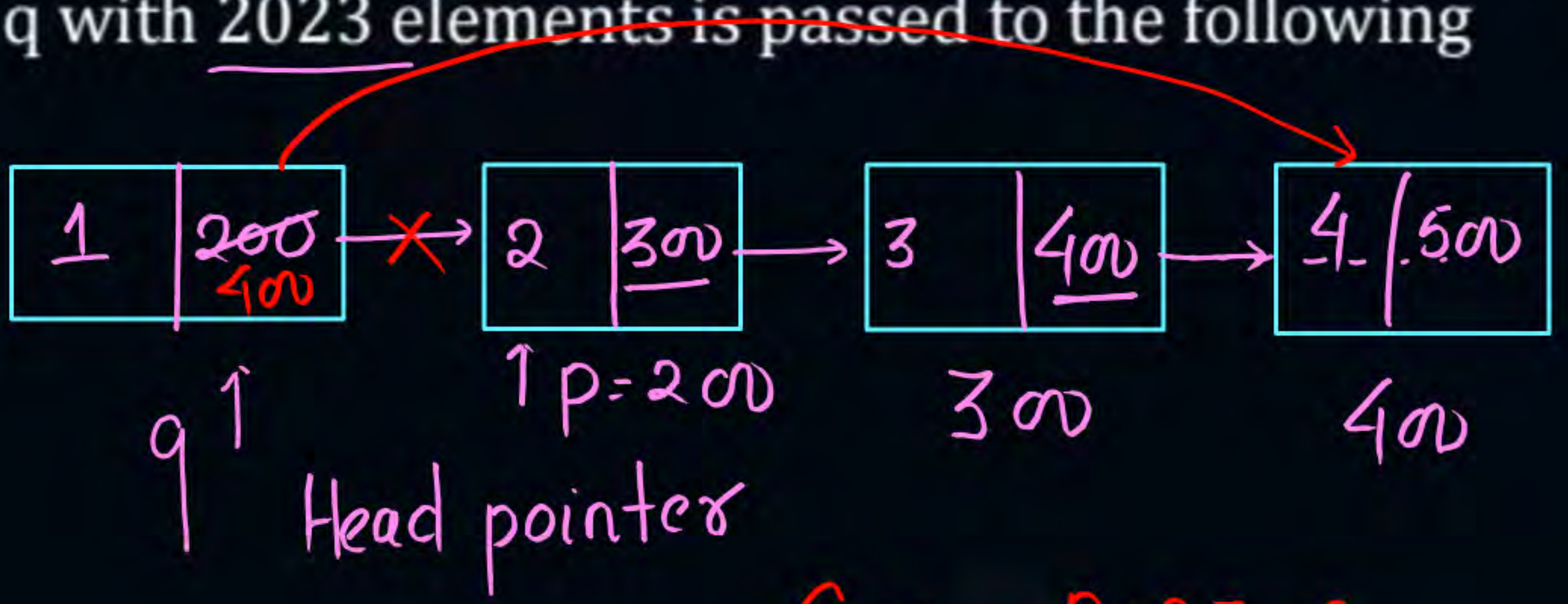
Discussion Notes

By- Abhishek Sir

[NAT]

#Q. 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;
}
```

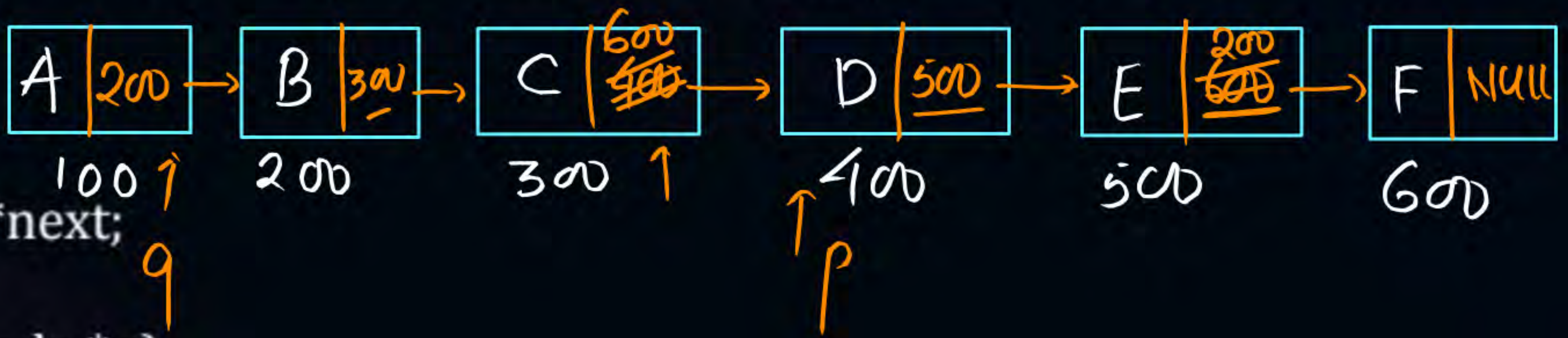


Size = 2023 - 2

The size of the linked list q after the execution of the function is 2021.

[MCQ]

#Q. 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;
         200      300      400
    p=q->next->next->next;
    q->next->next->next=p->next->next;
    p->next->next=q->next;
    printf("%c", p->next->next->next->data);
}
```

C

The output is-

[✓ A C]

B D

C E

D B

[MCQ]

#Q. Consider the following statements:

P: Linked Lists supports linear accessing of elements ✓

Q: Linked Lists supports random accessing of elements. ✗

Which of the following statements is/are INCORRECT?

[B]

A P only

B Q only ✓

C Both P and Q

D Neither P nor Q

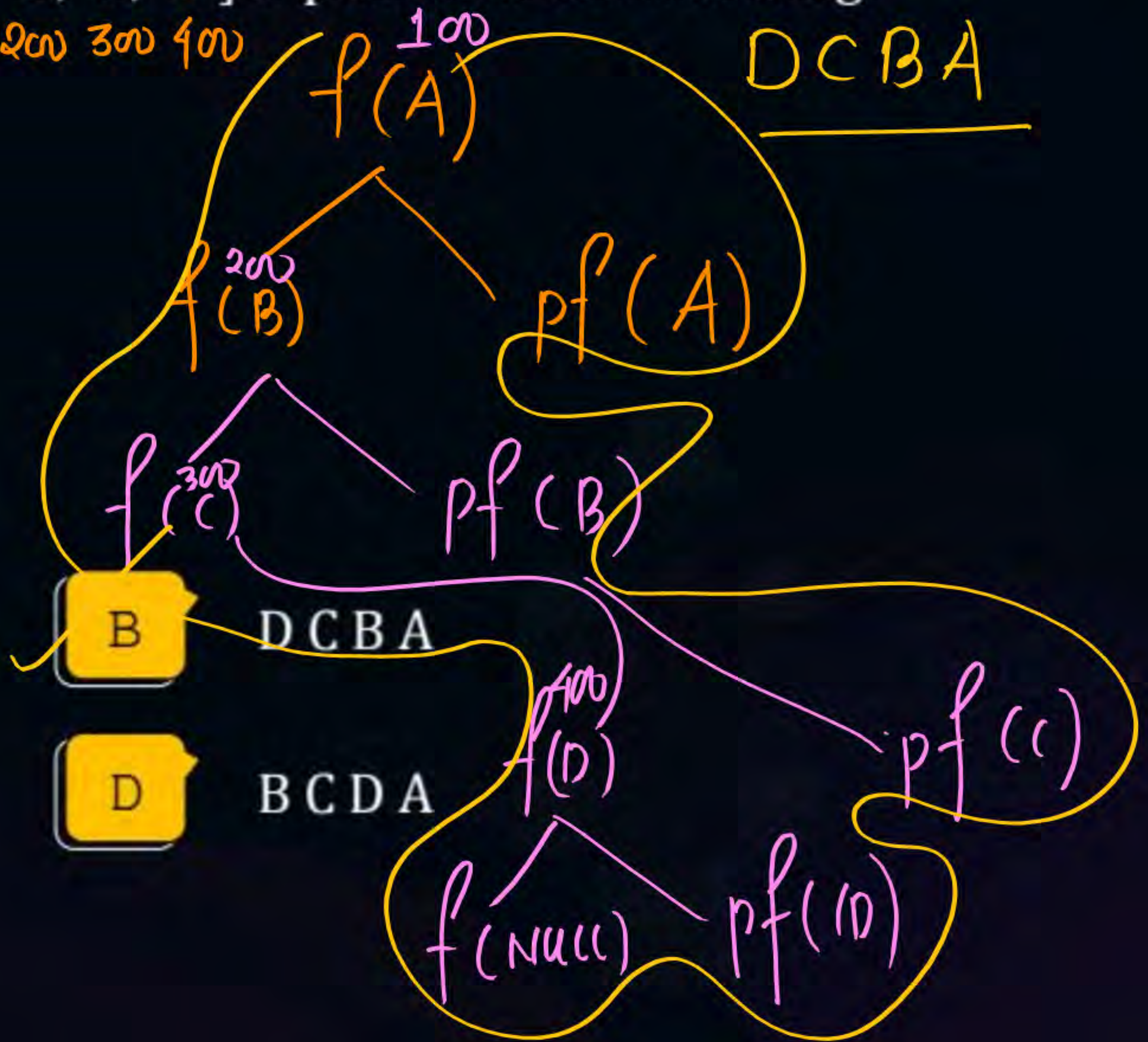
[MCQ]

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

100 200 300 400

```
void f(struct node *q)
{
    if(q==NULL) return;
     f(q->next);
     printf("%c ", q->data);
}
```

The output is-



A CDBA

[✓ B] DCBA

C ABCD

D BCDA

[MCQ]

#Q. Consider the following statements:

time Complexity

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? ✓

Traversal

[✓ A] Both P and Q

B P only

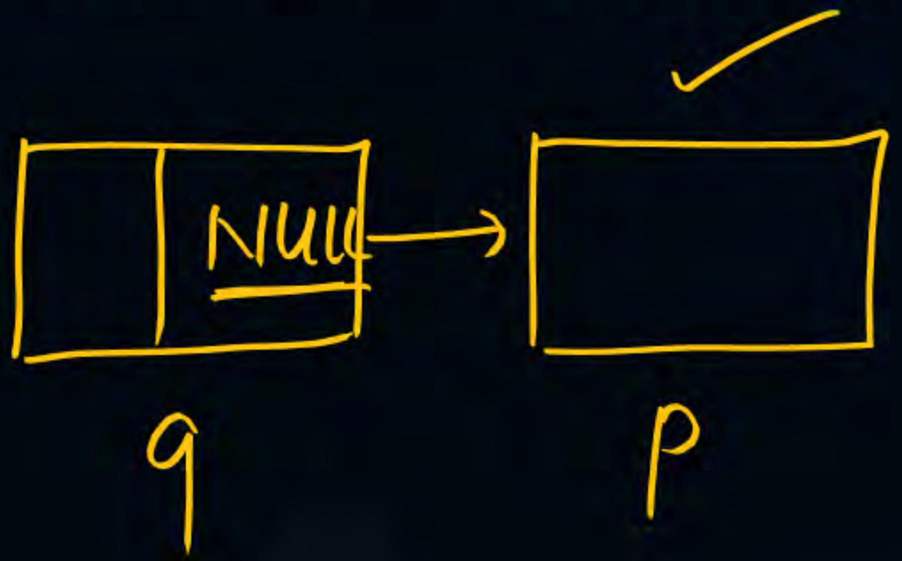
C Q only

D Neither P nor Q

[MCQ]

#Q. 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;}    p→next != NULL
   while(_____a________){
     q = p;
     p=p->next; ✓
   }
   _____b________;    q→next = NULL
free(p);
 q=NULL; p=NULL;
}
```



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

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

C a: p->next ! = NULL ; b: q->next = NULL ✓ [c]

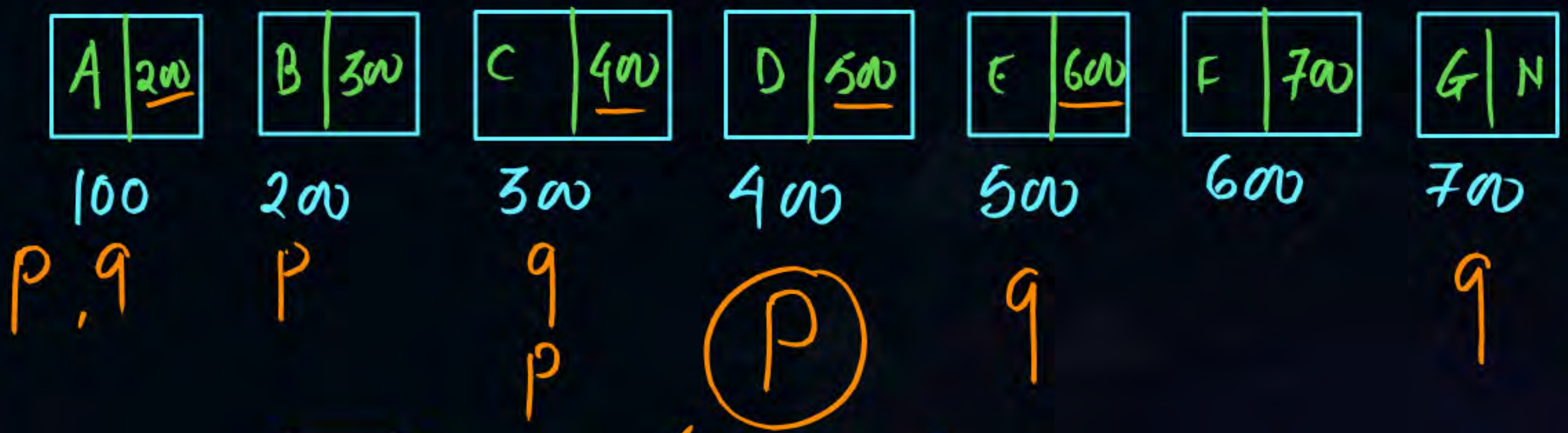
D a: head->next ! = p ; b: q->next = p

[MCQ]

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

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

The output is-



A. C

B. D ✓

[D]

C. E

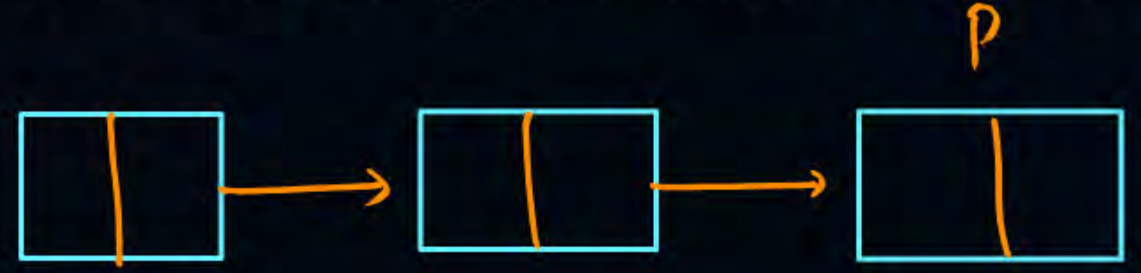
D. B

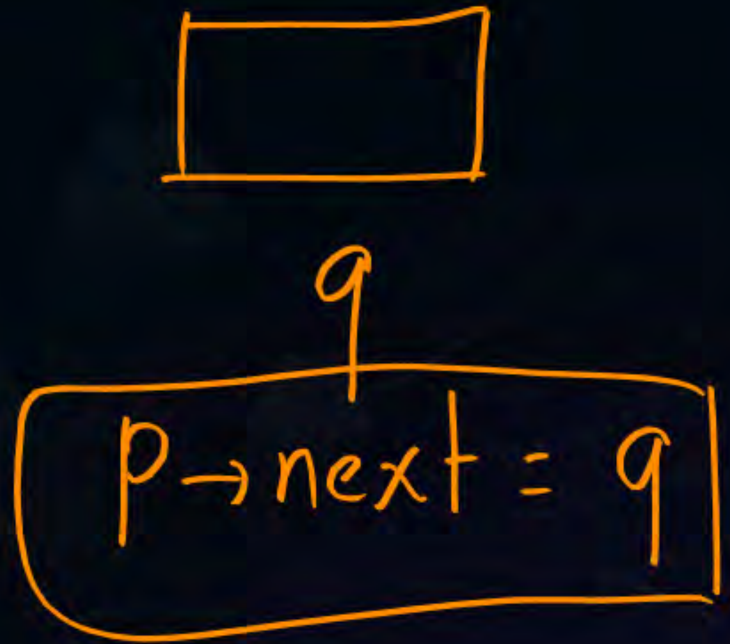
[MCQ]

#Q. 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;
         }
}
```

p→next != 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

THANK - YOU