CS & IT

GATE

Data Structure & Programming

DPP: 1

Linked List

Q1 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 _____.

Q2 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

Q3 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?

(A) P only

(B) Q only

(C) Both P and Q

(D) Neither P nor Q

Q4 Consider a single linked list q['A', 'B', 'C', 'D'] is passed to the following function:

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

The output is

(A) C D B A (B) D C B A

(C) A B C D (D) B C D A

Q5 Consider the following statements:

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

Q6 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;
}
```

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

Q7 Consider a single linked list q[['A', 'B', 'C', 'D', 'E', 'F', 'G'] is passed to the following function:

```
void func(struct node *head){
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
(C) E (D) B

Q8 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

Answer Key

Q1	2021~2020
Q2	
Q3	
Q4	
Q5	
Q6	
Q7	
Q8	

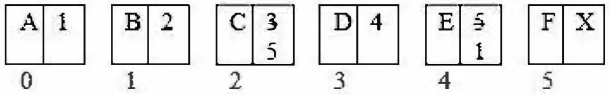
Hints & Solutions

Q1 Text Solution:

The above function implementation skip the second and third elements. It connects the head element to the fourth element.

So, the size of the linked list is 2021.

Q2 Text Solution:



X represents NULL.

Initially, q points to node 0.

p=q->next->next->next;//p=3

q->next->next->next=p->next->next;//2->next=5

p->next->next=q->next;//4->next=1

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

3->next->next->next->data

=4->next->next->data

=1->next->data

=2->data

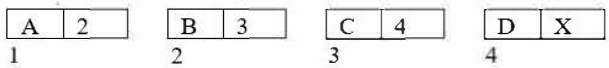
=C

Q3 Text Solution:

Linked List supports only linear accessing of elements.

Q4 Text Solution:

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



X represents NULL.

f(1):

1 is NOT NULL.

f(2):

P4: It prints 1->data i.e A.

f(2):

2 is NOT NULL.

f(3).

P3:

It prints 2->data i.e B.

f(3):

3 is NOT NULL

f(X).

P2:

It prints 3->data i.e C.

f(4):

4 is NOT NULL;

f(X).

P1:

It prints 4->data i.e D.

f(X):

X is equal to NULL. So it returns to f(4);

OUTPUT: D C B A

Q5 Text Solution:

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

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

Q6 Text Solution:

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

```
while(p->next!=NULL)
{
    q = p;
    p=p->next;
}
q->next=NULL;
free(p);
q=NULL; p=NULL;
}
```

Q7 **Text Solution:**

The code prints the middle element in the linked list q.

A B C D E F G.

Output: D

Q8 **Text Solution:**

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