Data structure & Programming

Linked List

DPP-01

[NAT]

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 _____.

[MCQ]

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

[NAT]

3. 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

[MCQ]

4. 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

[NAT]

5. 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

[NAT]

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;
}
```

(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

[MCQ]

7. 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

[NAT]

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

Answer Key

1. **(2021)**
2. **(a)**
3. **(b)**
4. **(b)**
5. **(a)**
6. **(c)**
7. **(b)**
8. **(c)**

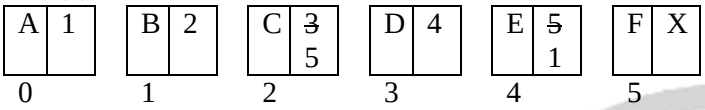
Hints and Solutions

1. (2021)

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.

2. (a)



X represents NULL.
Initially, q points to node 0.

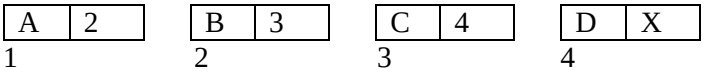
p=q->next->next->next;//p=3
q->next->next->next=p->next->next;//2->next=5
p->next->next=q->next;//4->next=1
printf("%c", p->next->next->next->data);
3->next->next->next->data
=4->next->next->data
=1->next->data
=2->data
=C

3. (b)

Linked List supports only linear accessing of elements.

4. (b)

```
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

5. (a)

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.

6. (c)

```
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;
}
```

7. (b)

The code prints the middle element in the linked list q.
A B C D E F G.
Output: D

8. (c)

```
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;
}
```