4
CHAPTER

Queue

Q.1 In breadth first search of graph, which data structure is used.

(a) Stack (b) Queue
(c) Array (d) Linked List

Q.2 If the number 4, 3, 2, 1 are placed in queue (in that order), and then removed one at a time in what order they will be removed?

Q.3 Let *q* be a queue and *S* be a stack. Assume that *q* and *S* are initially empty.

```
enqueue (q, 8);
enqueue (q, 3);
Push (S, 7);
Push (S, 9);
for (i = 0; i < 5; i++)
{
    printf ("%d", dequeue (q));
    printf ("%d", Pop (S));
    enqueue (q, i);
    Push (S, i + 5);
}
```

The last integer value printed by the following code is ________.

Q.4 Suppose the number 1, 2, 3, 4, 5 and 6 arrive in an input stream in that order. Which of the following sequences can be realized as the output of double ended queue?

(*i*) 1 2 3 4 5 6 (*ii*) 2 4 3 6 5 1
(*iii*) 6 5 4 3 2 1 (*iv*) 1 5 2 4 3 6

(a) (*i*),(*iv*) only (b) (*ii*), (*iii*) only
(c) (*i*) and (*ii*) only (d) All of these

Q.5 Consider the following function.

```
void madeeasy (int n)
{
    enqueue (Q, 0);
    enqueue (Q, 1);
    for (i = 0; i < n; i++)
    {
        x = dequeue (Q);
        y = dequeue (Q);
        enqueue (Q, y);
        enqueue (Q, x + y);
        print (x);
    }
}
```

What is the functionality of above function madeeasy?

(a) Prints numbers from 0 to n – 1
(b) Prints numbers from n – 1 to 0
(c) Prints first n fibbonacci numbers
(d) Prints first n fibbonacci numbers in reverse order

Q.6 Consider the following function.

```
find (Q)
{
    while (! Q. isEmpty ( ))
        S. push (Q. dequeue ( ));
    while (! S. isEmpty ( ))
        Q.enqueue (S. pop ( ));
}
```

Assume *S* is stack which is initially empty and *Q* is queue which contain *n*-elements. The functions:

(1) isEmpty( ) returns true if stack/queue is empty otherwise returns false.
(2) Push( ) and Pop( ) functions are standard stack operations.
(3) enqueue( ) and dequeue( ) are queue operations.

What will be the contents of *Q* after execution of find (*Q*)? (Assume S and *Q* are global variables)

(a) No change to the contents of *Q*
(b) *Q* will be empty (No element in the queue)
(c) *Q* will be reversed (All elements are in the reverse order)
(d) None of these

Q.7 A circular array based queue *q* is capable of holding 7 elements. After execution of the following code, find the element at index '1', if the array is initially empty and array has indices from 0 to 6.

```
for (x = 1; x <= 6; x++)
    q.enqueue (x);
for (x = 1; x <= 3; x++)
{   q.dequeue ( );
    q.enqueue (q.dequeue ( ));
}
```

Assume enqueue and dequeue are circular queue operations for insertion and deletion respectively.

Q.8 Let *q* be a queue and *S* be a stack. Assume that *q* and *S* are initially empty. What is the last integer value printed by the following code?

```
enqueue (q, 8);
enqueue (q, 3);
Push (S, 7);
Push (S, 9);
for (i = 0; i < 5; i++)
{
    printf ("%d", dequeue (q));
    printf ("%d", Pop (S));
    enqueue (q, i);
    Push (S, i + 5);
}
```

Q.9 Suppose that stacks and queues are provided as opaque data types, offering only operations to add elements, to remove elements, and to test for emptiness. Suppose that a programmer wants to count the number of elements in a given stack or queue *C*, which is currently in some state *t*, using only one auxiliary stack or queue *D*. The structures *C* and *D* can be used in any way possible based on the methods they offer, but *C* must be restored to its state t after counting its elements. Counting elements as described above is possible for which of the following data types?

1. *C* is a queue and *D* is a queue.
2. *C* is a stack and *D* is a stack.
3. *C* is a queue and *D* is a stack.

(a) None (b) 1 and 2 only
(c) 1 and 3 only (d) 1, 2, and 3

Q.10 Suppose you are given an implementation of a queue of integers the operations that can be performed on the queue is

1. empty (*Q*)- returns true if the queue empty, false characters
2. delete (*Q*) - deletes the element at the front of the queue and return its value.
3. insert (*Q*, *i*) - insert the integer *i* at the rear of queue.

Consider the following function

```
void f (queue Q)
{  int i;
   if (! isempty (Q))
   {  i = delete (Q);
      f(Q);
      insert (Q, i);
   }
}
```

What operation is performed by the above function *f*?

(a) Leaves the queue *Q* unchanged
(b) Reverse the order of element in the queue
(c) Deletes the element at the front of the queue *Q* and inserts it at the rear keeping the other elements in the same order
(d) Empties the queue (*Q*)

Q.11 Consider the following code :

```
void find( )
{
    tail = head;
    struct node * x = head;
    struct node * y;
    head = NULL;
    while(x! = NULL)
    {
        y = x → next;
        x → next = head;
        head = x;
        x = y;
    }
}
```

Assume a queue is implemented using linked list with head and tail pointers. If head points to the first element in queue and tail points to the recently inserted element. Find the functionality of "find" if head and tail are global variables.

(a) It traverses the queue
(b) It reverses the queue
(c) It searches a node in the queue
(d) It will keep the queue as same

Q.12 In an empty queue, rear and front can be initialized as
(a) rear = front = −1
(b) rear = front = 1
(c) rear = 0, front = −1
(d) rear = −1, front = 0

■■■■

Queue

1 ⇒ $\boxed{B}$ BFS of graph used ⟹ Queue

2 ⇒ $\boxed{4321}$ 4, 3, 2, 1 in queue, removing order —

$\rightarrow$ 4, 3, 2, 1 Ans

3 ⇒ $\boxed{8}$ O/P = 8, 9, 3, 5, 0, 6, 1, 7, 2, 8

last O/P

(4)

I/P → 1, 2, 3, 4, 5, 6

(R) O/P in double ended queue - ?

1, 2, 3, 4, 5, 6 ✓
2, 4, 3, 6, 5, 1 ✓
6, 5, 4, 3, 2, 1 ✓
1, 5, 2, 4, 3, 6 ✓

All

(5)

(C)

```
void madeeasy (int n)
{
      enqueue( Q, 0);
      enqueue( Q, 1);
     for ( i = 0; i < n; i++)
      {     x = dequeue (Q );
            y = dequeue ( Q);
            enqueue ( Q, y);
            enqueue ( Q, x+y);
            print(x);
      }
}
```

Print 1st n fibbonacci numbers Ans

6 →

(2)

```
find(Q)
{ while( ! Q.isEmpty())
        S.push( Q.dequeue());
  while ( ! S.isEmpty())
        Q.enqueue(S.pop());
}
```

Let
S → Stack
Q → Queue

Q will be reversed (All elements in reversed order)

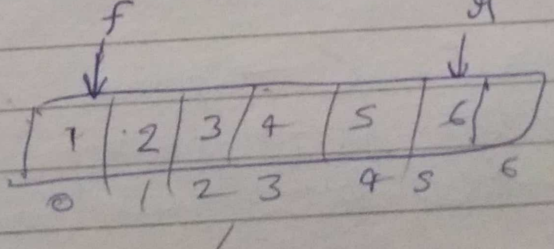
Ans

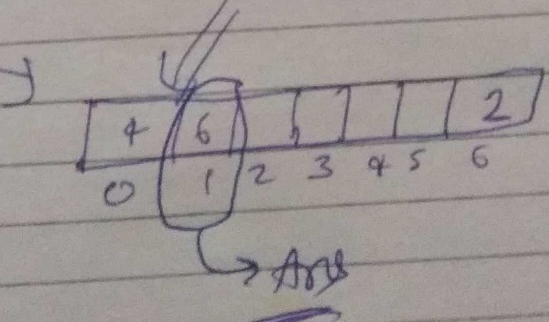
7 →

(8)

```
for ( x =1 ; x<=6 ; x++)
        q.enqueue(x);
for (x = 1; x <= 3; x++)
    { q.dequeue();
        q.enqueue(q.dequeue());
    }
```

→ circular queue





(8)⇒ $\boxed{8}$

```
enqueue ( q,8);
enqueue (q, 3);
push (S, 7);
push (S, 9);
for (i=0 ; i<5 ; i++)
{
    printf("%d", dequeue(q));
    printf("%d", pop(S)),
    enqueue ( q, i);
     push ( S, i+5);
}
```

O/P ⇒ 8, 9, 3, 5, 0, 6, 1, 7, 2, (8)

last o/p

Ans

(9)⇒ D

Counting elements as

→ (1) C → queue , D → queue

→ (2) C → stack , D → Stack } All

→ (3) C → queue , D → Stack

10 B

```
void f(queue Q)
{   int i;
    if(!isempty(Q))
    {   i = delete(Q);
        f(Q);
        insert(Q,i);
    }
}
```

Reverse the order of queue

Ans

11 B

```
void find()
{ tail = head;
  struct node *x = head;
  struct node *y;
  head = NULL;
  while(x != NULL)
  { y = x->next;
    x->next = head;
    head = x;
    x = y;
  }
}
```

Reverse the queue

Ans

12 A

Empty queue ⟹ rear = front = -1

Ans