OVERALL ANALYSIS Solution Report

All Correct Answers Wrong Answers Not Attempted Questions

Q.1) Max Marks: 1

Choose the option, which is not correct for priority queue.

A **In priority queue, every item has a priority associated with it.**

B **An element of priority queue with low priority is de-queued before an element with high priority.** Correct Option

Solution: (B)
Solution: Ans is (B).
In priority queue, every item has a priority associated with it. : yes it is true.
An element of priority queue with low priority is de-queued before an element with high priority. : not correct
If two elements have the same priority, they are served according to their order in the queue. : correct
A priority queue is typically implemented using Heap data structure. : correct

Option (B) is not correct because an element of priority queue of high priority is dequeued before an element with low priority. And all the other statements are the properties of priority queue.

C **If two elements have the same priority, they are served according to their order in the queue.**

D **A priority queue is typically implemented using Heap data structure.**

Q.2) Max Marks: 1

In the given singly linked list, a pointer is pointing to a node x in this list. There is no head pointer in this singly linked-list. We are performing delete operation on a random node using pointer x. When the delete operation is failing ?

A **If size of linked-list is even.**

B **If node is the next of x.**

C **If node is previous of x.** Correct Option

Solution: (C)

Solution: Ans is (C). If node is previous of x.

It would be a simple deletion problem from the singly linked list if the head pointer was given because for deletion you must know the previous node and you can easily reach there by traversing from the head pointer. Conventional deletion is impossible without knowledge of the previous node of a node which needs to be deleted.

Here, if we want to delete previous node of where x is pointing. At that time deletion will be failed so the correct answer is " if node is previous of x".

D **If x is pointing to the first node.**

Q.3) Max Marks: 1

When two strings s1 and s2 are given of size n1 and n2 and concatenation of these two strings are performed with the help of Circular doubly linked list. Then what will be time complexity of for this.

A **O(1)** Correct Option

Solution: (A)
Solution: O(1)
Given that
two strings s1 and s2 of size n1 and n2 are given.
suppose s1 = "aaic" with initial address 1000 and s2 = "gate" with initial address 2000
after putting s1 string in the circular doubly linked list, we can easily get the address of s1 last char and s2 initial address so with the help of circular double linked list we can find out the last node of the first string in O(1) time because head will have the reference of the last node and then we will add second string here. So overall time complexity will be O(1).

B **O(log n) if total characters are n.**

C **O(n) if total characters are n.**

D **O(n^2) if total charachters are n**

Q.4) Max Marks: 1

If a singly linked-list with 33 elements is given then what will be the time complexity of getting 11th element from the starting and 23rd element from the last.

A O(n) and O(1)

B O(1) and O(n)

C O(1) and O(1) Correct Option

Solution: (C)
Solution: Ans is (C). O(1) and O(1)
A singly linked list with 33 elements is given. Then
getting 11th element from starting: Time complexity will be O(n) where n is the length of the list. In the given question length is 33 so accessing 11th element then time complexity = O(11) = O(1).
getting 23rd element from last: Time complexity will be O(n) where n is the length of the list. In the given question total length is 33 so 23rd from last means 11th from starting so accessing 11th element will take O(11) = O(1) time.
As here, elements in the singly linked list are fixed so time complexity will be O(1) in all cases

D O(n) and O(n)

Q.5) Max Marks: 1

Which of the following permutation is correct for inserting into the stack for getting the output 9, 6, 2, 3, 4, 1 (in the same order) using stack data structure.

A 4, 2, 3, 9, 6, 1

B 2, 3, 9, 1, 6, 4

C 1, 2, 3, 4, 6, 9

D 9, 6, 4, 3, 2, 1 Correct Option

Solution: (D)
Solution: Ans is (D). 9, 6, 4, 3, 2, 1
Here, we are taking option d as input. Then first we do
Push (9) and pop
Push (6) and pop
Push (4), push (3), push (2), pop, pop, pop
Push (1) and pop
So we will get sequence 9, 6, 4, 3, 2, 1
Check other options by using the above procedure.

Q.6) Max Marks: 1

There are some statements are given choose the correct word for them.
S1. For infix to postfix conversion
S2. For postfix evaluation
S3. For infix to prefix conversion
(a). Operator Stack
(b). Operand stack
(c). time complexity O(n)

A S1 - (a), S2 - (b), S3 - (c) Correct Option

Solution: (A)
Solution: Ans is (A). S1 - (a), S2 - (b), S3 - (c)
Here, we need to make one to one matches and we can not left any.
As we use Operator Stack for infix to postfix conversion and Operand stack for postfix evaluation. And time complexity for infix to prefix conversion is O(n). Correct matches are as follows:
S1. For infix to postfix conversion -- (a) operator stack
S2. For postfix evaluation -- (b) operand stack
S3. For infix to prefix conversion -- (c) time complexity O(n)

B S1 - (c), S2 - (b), S3 - (a)

C S1 - (a), S2 - (c), S3 - (b)

D S1 - (b), S2 - (a), S3 - (c)

Q.7) Max Marks: 1

A lower triangular matrix X[0 - - m] Y[0 - -m] is given where m=79. This matrix is stored in the array of n size and this array already has 90 elements in it(i.e these elements are not part of the given matrix). So we will start to store elements after this. Then what will be the value of n for which all matrix elements can be stored in this array sufficiently.

A 3160

3220 Correct Option

B 3330 Correct Option

Solution: (B)
Solution: 3330
GIven that the matrix is Lower triangular Matrix,
example of lower triangular matrix of size 3*3:
1 0 0
2 3 0
4 5 6

therefore 80*80 matrix number of non zero elements are
1 for row 0
2 for row 1
3 for row 2
.
.
.
80 for row 79
==> Total number of non zero elements are 1 + 2 + 3 + ... + 79 +80 = (80*81)/2 = 3240
Given Array already has 90 elements apart from matrix elments = 3240 + 90 = 3330
best possible storage technique adopted here is storing the non-zero elements in Row Major Order.

C 3240

D 3171

Q.8) Max Marks: 1

Let Q denote a queue containing thirteen numbers and S be an empty stack.
Head (Q) returns the element at the head of the queue Q without **removing it from Q.**
Top(S) returns the element at the top of S without **removing it from S.**
Both the cases are utilized until queue gets empty.
(1). we are dequeuing and pushing elements in stack if stack is empty or top(S) is < = head(Q)
(2). we are popping and enqueuing if top(S) is > head(Q)

Choose the correct answer for number of times this will be done for the worst case.

A 13

B 130

C 143

D 169 Correct Option

Solution: (D)
Solution: Ans is (D) 169
A queue is given with 13 elements.
head(Q): return element from head of queue without removing it.
if top(S) < = head(Q) : dequeuing and pushing
if top(S) > head(Q) : popping and enqueuing
with these given conditions, worst case will be when queue Q is sorted in descending order.
That time there will be n*n condition check for making queue empty.
So, here 13* 13= 169 times.

Q.9) Max Marks: 1

When an expression is given like this

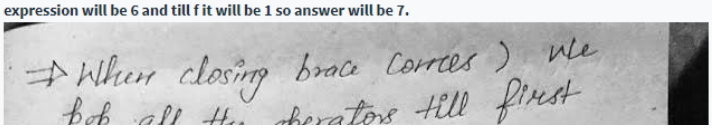
a↑ (b-c) * (d/(e+f)) ↑ (g * (h-i))

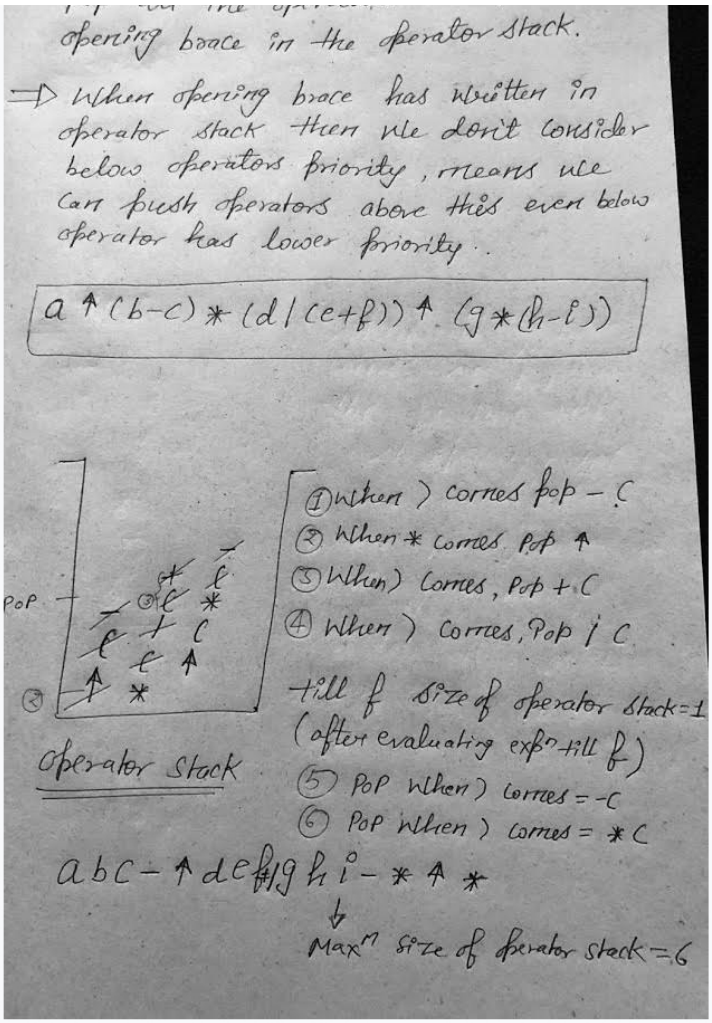
While writing its postfix expression using operator stack, then what will be the sum of maximum size of the operator stack and size of operator stack after evaluating expression till f.

A 6

B 7 Correct Option

Solution: (B)
Solution: Ans is (B) 7
Here, concept is to pop operator from the operator stack when low priority operator come over the high priority operator. So after evaluating expression highest size of expression will be 6 and till f it will be 1 so answer will be 7.





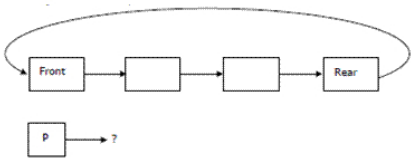
So total size will be 6+1 = 7

C 8

D 9

Q.10) Max Marks: 1

A queue 'Q' is behaving as circular linked list. A single variable 'p' is used for accessing the queue. Then from rear node of queue enQueue and deQueue operation are performed.



Then calculate the time in which operations would have been done.

A **constant time** Correct Option

Solution: (A)

Solution: Ans is (A). constant time
Here, if we do enqueue and dequeue from the rear of the linked list then it will take constant time only.

```
for enqueue:

newnode -> next = rear ->next;

rear->next = newnode;

rear = newnode;

for dequeue:

struct node* front;

front = rear-> next;

rear->next = front->next;
```

```
free(front);
```

B logarithmic time

C polynomial time

D None of these

Q.11) Max Marks: 2

A matrix m[30][30] is stored in an array which starting with address 1000 in row major order. An element x is stored at address x[i][j]= 1439 and j is given as 20 then find the value of i while arrays are starting from 1.

A 10

B 14

C 15 Correct Option

Solution: (c)

Solution: Ans is (C).15
Given that, x[i][j]= 1439
for getting location of an element which is stored at x[i][j] location while storing elements in row major order.
= base address + (i-1) *30 +(j-1)
=> 1439 = 1000 + (i-1)*30 + (20-1)
= 1000 + 30i -30 +19 = 30i + 989
30i =1439-989 = 450 => i = 15

D 17

Q.12) Max Marks: 2

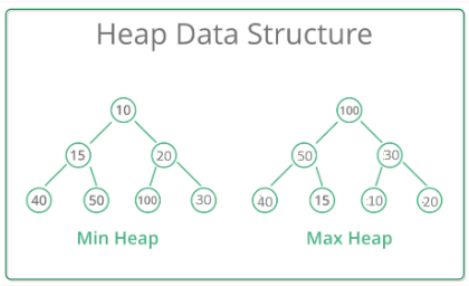
An array of elements is given as 10, 15, 30, 40, 50, 100, 20. Then construct min-heap and max-heap of the given elements. Find out how many elements are still at the same node in both the heaps. Please note: heapify is being applied after inserting all the elements while constructing min/max heap (complete binary tree)

A 3

B 2

C 1 Correct Option

Solution: (c)
Solution: Ans is (C) 1.
Heap: A Heap is a data structure in which the tree is a complete binary tree. Generally, Heaps can be of two types:
Max-Heap: In a Max-Heap the key present at the root node must be greatest among the keys present at all of its children. The same property must be recursively true for all sub-trees in that Binary Tree.
Min-Heap: In a Min-Heap the key present at the root node must be minimum among the keys present at all of its children. The same property must be recursively true for all sub-trees in that Binary Tree.



There is only one element 40, which is at the same position in both the heaps so answer will be 1.

D 4

Q.13) Max Marks: 2

A linked-list S is given below and a pseudo code is written in which last node of the linked list is moved to the front of the list. Just choose the order of statements if operation is performed successfully.

S: A (2000) -> B (3000) -> C (4000) -> D (5000) -> E (NULL)

P=S;

```
While (P-> next ! = NULL)
{
  Q=P;
  P= P-> next ;
}
```

(i) _____

(ii) _____

(iii) _____

1. S = P;
2. Q -> next = NULL;
3. P-> next = S;

A (i)-2, (ii)-3, (iii)-1 — Correct Option

Solution: (A)
Solution: Ans is (A). (i)-2, (ii)-3, (iii)-1
Order is really important for executing that particular operation of linked-list. As first statement is storing linked list. Second statement we are making next of first element of Q as NULL. And now we are assigning next of P as S. Basically we are doing this as:
P=S; // first we are storing single linked in another variable P
while (P--> next != NULL) // when we get last element's next as NULL then loop will be terminated
{
Q = P // just one step before termination, we are storing second last node in the variable Q.
P= P--> next // everytime it is moving forward while pointing p to p--> next.
}
Q-->next = NULL ; // making second last variable's next as Null because after removal of last element it will become last and its next must be null.
P--> next = S; // making next of P as S so first element of S will become next element as we are adding last element as first.
S = P; // now S will start pointing to the P, and P is the first element of the list.

B (i)-1, (ii)-3, (iii)-2

C (i)-1, (ii)-2, (iii)-3

D None of these

Q.14) Max Marks: 2

An empty queue is given in which 5 elements are going to be inserted then for enqueue it is taking 3 sec and for dequeue it is taking 3 seconds. And there is time elapse between enqueue- enqueue and dequeue- dequeue of 1 sec. And for enqueue-dequeue is of 2 seconds. Then calculate the total time of all the 5 elements (from enqueuing all elements to dequeuing all).

A 30 sec

B 39 sec

C 40 sec — Correct Option

Solution: (C)
Solution: Ans is (C). 40 sec

Here, first we enqueue all the elements in the queue and then all the elements are dequeued from the queue.

For enqueue 5 elements = 3*5 = 15 , 4 gaps = 4 elapsed time
For enqueue- dequeue =2 because just after inserting 5th element, we are performing dequeuer.
For dequeue 5 elements = 3 *5 = 15, 4 gaps = 4 elapsed time
So total time = 15+4+2+15+4 = 40 sec

D 41 sec

Q.15) Max Marks: 2

Consider the following function that takes reference to head of a Doubly Linked List as parameter. Assume that a node of doubly linked list has previous pointer as *prev* and next pointer as *next*.

```
void fun(struct node **head_ref)
{
  struct node *temp = NULL;
  struct node *current = *head_ref;

  while (current != NULL)
```

```
while (current !=  NULL)
 {
   temp = current->prev;
   current->prev = current->next;
   current->next = temp;
   current = current->prev;
 }

    if(temp != NULL )
   *head_ref = temp->prev;
}
```

Assume that reference of head of following doubly linked list is passed to above function 1 <--> 2 <--> 3 <--> 4 <--> 5 <-->6. What should be the modified linked list after the function call?

A. 1 <--> 2 <--> 3 <--> 4 <--> 5 <-->6

B. 5 <--> 4 <--> 3 <--> 4 <--> 1 <-->6

C. 6 <--> 5 <--> 4 <--> 3 <--> 2 <-->1 — Correct Option

Solution: (C)

Solution: Ans is (C). 6 <--> 5 <--> 4 <--> 3 <--> 2 <-->1

from the given code: The given function reverses the given doubly linked list.

//function to reverse the doubly linked list.

```
void fun(struct node **head_ref)
        {
       struct node *temp = NULL;
       struct node *current = *head_ref;
```

// swap next and prev for all nodes of doubly linked list.

```
       while (current !=  NULL)
      {
          temp = current->prev;
          current->prev = current->next;
          current->next = temp;
          current = current->prev;
        }
```

// before changing head, check for the cases empty list and list with only one node.

```
           if(temp != NULL )
          *head_ref = temp->prev;
         }
```

D. 6 <--> 5 <--> 4 <--> 1 <--> 2 <-->3

close