

# DSA Lab Mid Term Test

## Duration 60 minutes.

Name:


ID:

Your answer:

1.	2.	3.	4.	5.	6.	7.	8.	9.	10.
11.	12.	13.	14.	15.	16.	17.	18.	19.	20.

Q1.




Following is C like pseudo code of a function that takes a Queue as an argument, and uses a stack S to do processing.



```
void fun(Queue *Q)
{
    Stack S; // Say it creates an empty stack S

    // Run while Q is not empty
    while (!isEmpty(Q))
    {
        // deQueue an item from Q and push the dequeued item to S
        push(&S, deQueue(Q));
    }

    // Run while Stack S is not empty
    while (!isEmpty(&S))
    {
        // Pop an item from S and enqueue the popped item to Q
        enqueue(Q, pop(&S));
    }
}
```



What does the above function do in general?

- (A)** Removes the last from Q
- (B)** Keeps the Q same as it was before the call
- (C)** Makes Q empty
- (D)** Reverses the Q

Q2.

Which one of the following is an application of Queue Data Structure?

- (A)** When a resource is shared among multiple consumers.
- (B)** When data is transferred asynchronously (data not necessarily received at same rate as sent) between two processes
- (C)** Load Balancing
- (D)** All of the above

Q3.

How many stacks are needed to implement a queue. Consider the situation where no other data structure like arrays, linked list is available to you.

- (A)** 1
- (B)** 2
- (C)** 3
- (D)** 4

Q4.

Which of the following is true about linked list implementation of queue?

- (A)** In push operation, if new nodes are inserted at the beginning of linked list, then in pop operation, nodes must be removed from end.
- (B)** In push operation, if new nodes are inserted at the end, then in pop operation, nodes must be removed from the beginning.
- (C)** Both of the above
- (D)** None of the above





Q4.

Suppose a circular queue of capacity  $(n - 1)$  elements is implemented with an array of  $n$  elements. Assume that the insertion and deletion operation are carried out using REAR and FRONT as array index variables, respectively. Initially,  $\text{REAR} = \text{FRONT} = 0$ . The conditions to detect queue full and queue empty are

- (A) Full:  $(\text{REAR} + 1) \bmod n == \text{FRONT}$ , empty:  $\text{REAR} == \text{FRONT}$
- (B) Full:  $(\text{REAR} + 1) \bmod n == \text{FRONT}$ , empty:  $(\text{FRONT} + 1) \bmod n == \text{REAR}$
- (C) Full:  $\text{REAR} == \text{FRONT}$ , empty:  $(\text{REAR} + 1) \bmod n == \text{FRONT}$
- (D) Full:  $(\text{FRONT} + 1) \bmod n == \text{REAR}$ , empty:  $\text{REAR} == \text{FRONT}$

Q5.

An implementation of a queue Q, using two stacks S1 and S2, is given below:

```
 void insert(Q, x) {  
    push (S1, x);  
}  
  
 void delete(Q){  
    if(stack-empty(S2)) then  
         if(stack-empty(S1)) then {  
            print("Q is empty");  
            return;  
        }  
        else while (!(stack-empty(S1))){  
            x=pop(S1);  
            push(S2,x);  
        }  
        x=pop(S2);  
}
```

Q6.

Let  $n$  insert and  $m$  ( $\leq n$ ) delete operations be performed in an arbitrary order on an empty queue  $Q$ . Let  $x$  and  $y$  be the number of push and pop operations performed respectively in the process. Which one of the following is true for all  $m$  and  $n$ ?

- (A)  $n+m \leq x < 2n$  and  $2m \leq y \leq n+m$
- (B)  $n+m \leq x < 2n$  and  $2m \leq y \leq 2n$
- (C)  $2m \leq x < 2n$  and  $2m \leq y \leq n+m$
- (D)  $2m \leq x < 2n$  and  $2m \leq y \leq 2n$

Explain the functionality of the following functions.

```
int fun1(int x, int y)
{
    if (x == 0)
        return y;
    else
        return fun1(x - 1, x + y);
}
```

Q7.

Predict the output of the following program.

```

#include <iostream>
using namespace std;

void fun(int x)
{
    if(x > 0)
    {
        fun(--x);
        cout << x << " ";
        fun(--x);
    }
}

int main()
{
    int a = 4;
    fun(a);
    return 0;
}

```

Q8.

Explain the functionality of below recursive functions.

```

void fun1(int n)
{
    int i = 0;
    if (n > 1)
        fun1(n - 1);
    for (i = 0; i < n; i++)
        cout << " * ";
}

```

Q9. Predict the output of the following program. What does the following fun() do in general?

```

#include <iostream>
using namespace std;
int fun(int a, int b)
{
    if (b == 0)
        return 0;
    if (b % 2 == 0)
        return fun(a + a, b/2);

    return fun(a + a, b/2) + a;
}

int main()
{
    cout << fun(4, 3) ;
    return 0;
}

```

Q10.

What does the following function do for a given Linked List with first node as *head*?

```
void fun1(struct node* head)
{
    if(head == NULL)
        return;

    fun1(head->next);
    printf("%d  ", head->data);
}
```

- (A) Prints all nodes of linked lists
- (B) Prints all nodes of linked list in reverse order
- (C) Prints alternate nodes of Linked List
- (D) Prints alternate nodes in reverse order

Q11.

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*.

The following function `reverse()` is supposed to reverse a singly linked list. There is one line missing at the end of the function.

```

/* Link list node */
struct node
{
    int data;
    struct node* next;
};

/* head_ref is a double pointer which points to head (or start) point
of linked list */
static void reverse(struct node** head_ref)
{
    struct node* prev = NULL;
    struct node* current = *head_ref;
    struct node* next;
    while (current != NULL)
    {
        next = current->next;
        current->next = prev;
        prev = current;
        current = next;
    }
    /*ADD A STATEMENT HERE*/
}

```

What should be added in place of "/\*ADD A STATEMENT HERE\*/", so that the function correctly reverses a linked list.

- (A) \*head\_ref = prev;
- (B) \*head\_ref = current;
- (C) \*head\_ref = next;
- (D) \*head\_ref = NULL;

Q12.



```

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

    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) 2 1 4 3 6 5
- (B) 5 4 3 2 1 6.
- (C) 6 5 4 3 2 1.
- (D) 6 5 4 3 1 2

Q13. What is the output of following function for start pointing to first node of following linked list?

1->2->3->4->5->6

```

void fun(struct node* start)
{
    if(start == NULL)
        return;
    printf("%d ", start->data);

    if(start->next != NULL )
        fun(start->next->next);
    printf("%d ", start->data);
}

```

**(A)** 1 4 6 6 4 1

**(B)** 1 3 5 1 3 5

**(C)** 1 2 3 5

**(D)** 1 3 5 5 3 1

Q14. The following C function takes a simply-linked list as input argument. It modifies the list by moving the last element to the front of the list and returns the modified list. Some part of the code is left blank. Choose the correct alternative to replace the blank line.

```
typedef struct node
{
    int value;
    struct node *next;
}Node;
```

```
Node *move_to_front(Node *head)
{
    Node *p, *q;
    if ((head == NULL: || (head->next == NULL))
        return head;
    q = NULL; p = head;
    while (p-> next !=NULL)
    {
        q = p;
        p = p->next;
    }
    _____
    return head;
}
```

- (A)** q = NULL; p->next = head; head = p;
- (B)** q->next = NULL; head = p; p->next = head;
- (C)** head = p; p->next = q; q->next = NULL;
- (D)** q->next = NULL; p->next = head; head = p;

Q15.

What are the time complexities of finding 8th element from beginning and 8th element from end in a singly linked list?

Let  $n$  be the number of nodes in linked list, you may assume that  $n > 8$ .

- (A)**  $O(1)$  and  $O(n)$
- (B)**  $O(1)$  and  $O(1)$
- (C)**  $O(n)$  and  $O(1)$
- (D)**  $O(n)$  and  $O(n)$

Q16.

Given pointer to a node  $X$  in a singly linked list. Only one pointer is given, pointer to head node is not given, can we delete the node  $X$  from given linked list?

- (A)** Possible if  $X$  is not last node. Use following two steps (a) Copy the data of next of  $X$  to  $X$ . (b) Delete next of  $X$ .
- (B)** Possible if size of linked list is even.
- (C)** Possible if size of linked list is odd
- (D)** Possible if  $X$  is not first node. Use following two steps (a) Copy the data of next of  $X$  to  $X$ . (b) Delete next of  $X$ .

Q17.

You are given pointers to first and last nodes of a singly linked list, which of the following operations are dependent on the length of the linked list?

- (A)** Delete the first element
- (B)** Insert a new element as a first element
- (C)** Delete the last element of the list
- (D)** Add a new element at the end of the list

Q18.


Level of a node is distance from root to that node. For example, level of root is 1 and levels of left and right children of root is 2. The maximum number of nodes on level  $i$  of a binary tree is

In the following answers, the operator '^' indicates power.

- (A)  $2^{(i-1)}$
- (B)  $2^i$
- (C)  $2^{(i+1)}$
- (D)  $2^{[(i+1)/2]}$

19.

Following is C like pseudo code of a function that takes a number as an argument, and uses a stack S to do processing.



```
void fun(int n)
{
    Stack S; // Say it creates an empty stack S
    while (n > 0)
    {
        // This line pushes the value of n%2 to stack S
        push(&S, n%2);

        n = n/2;
    }





    // Run while Stack S is not empty
    while (!isEmpty(&S))
        printf("%d ", pop(&S)); // pop an element from S and print it
}
```

What does the above function do in general?

- (A) Prints binary representation of n in reverse order
- (B) Prints binary representation of n
- (C) Prints the value of Logn
- (D) Prints the value of Logn in reverse order

Q20.

Consider the following pseudocode that uses a stack

```
 declare a stack of characters
 while ( there are more characters in the word to read )
{
    read a character
    push the character on the stack
}
 while ( the stack is not empty )
{
    pop a character off the stack
    write the character to the screen
}

```

What is output for input "geeksquiz"?

- (A) geeksquizgeeksquiz
- (B) ziuqskeeg
- (C) geeksquiz
- (D) ziuqskeegziuqskeeg

