SHIV NADAR
INSTITUTION OF EMINENCE DEEMED TO BE UNIVERSITY
DELHI NCR

Programme: **B.Tech**
Exam**: Mid Semester**
Course Code: **CSD102**
Date: **March 04, 2025**

Discipline: **Computer Science and Engineering**
Year: **2024-2025**
Course Title: **Data Structures**
Time: **03:00 PM – 04:30 PM** Max. Marks: **50**

Name: _____ Roll Number: _____

Instructions:

Attempt Q1 to Q5 in the Question Paper and Remaining Questions (Q6 to Q9) in the provided Answer Sheets.

Ques.1 Indicate whether the following statements are True/False (No explanation is required): (5 Marks)

(a) A self-referential structure contains a pointer member that points to an object of a different type. **False**
(b) Creating and maintaining dynamic data structures requires dynamic memory allocation—a program's ability to obtain more memory space at execution time to hold new nodes and to release space no longer needed. **True**
(c) The following statement evaluates sizeof(struct node) to determine the size in bytes of a structure of type struct node, allocates sizeof(struct node) bytes in memory and stores a pointer to the allocated memory in variable newPtr. **True**

newPtr = malloc(sizeof(struct node));

(d) Function malloc returns a pointer of type void * to the memory it allocates. If it is unable to allocate memory, it returns a NULL pointer. **True**
(e) push is used to place elements on the bottom of a stack and pop is used to remove elements from the top of a stack. **False**

Ques.2 Choose the single correct option for the below questions: (14 Marks)

1.
```
void arrange (struct Node *head)
{
        if (head == NULL)
                return;
        arrange(head->next);
        free(head);
}
```

When applied on a linked list, the above code will

(a) Arrange the nodes in sorted order
(b) **Delete the complete list**
(c) Delete the head node of the list
(d) Arrange the nodes in sorted manner and delete the first node

2. When Binary Search is applied on a sorted array containing 26,000 elements, the maximum number of searches required would be

(a) 20 **(b)** **15** **(c)** 12 **(d)** 26000

3. What would the following function do when applied to a singly linked list?

```
void display(struct node *head)
  {         printf("%d    ", head->a);
            if (head->next == NULL)
                    return;
            display(head->next);          }
```

(a) Prints the list in reverse order
(b) prints only the data of the head node
(c) fails to print anything at all as head goes to NULL before display is called
(d) Prints the list in first to the last order

4. Given the following function tick the correct recurrence relation for T(n)

```
int debit ( int A[ ], int n)
{       if (n == 0)
                return 0;
        else
                return  A[n – 1] + debit(A, n – 1) + debit(A, n – 1);          }
```

(a) 2T(n – 1) + 2 **(b)** T(n – 1) + 2 **(c)** 3T(n – 1) + 2 **(d)** nT(n – 1)

5. A company records data for M employees using both array data structure and linked list data structure. N data values are associated with each employee. The record is sorted in terms of employee ID. An employee leaves the company. The worst time complexity for modifying the records would be

(a) O(N) for linked list and O(log N) for array
(b) O(M) for array and O(NM) for linked list
(c) O(log M) for array and O(M) for linked list
(d) O(N*M) for array and O(M) for linked list

6. The correct sequence in increasing order of time complexity would be

(a) O(log n) , O(√n), O(n), O(n log n), O(n^2)
(b) O(log n) , O(√n), O(n), O(n^2), O(n log n)
(c) O(1) , O(√n), O(log n) , O(n), O(n^2)
(d) O(log n) , O(n), O(2n), O(n^2)

7. What will the following function do when applied on a singly linked list

```
void fun1(struct Node *head)
{       struct Node* temp = head;
        while (temp != NULL)
        {       printf("%d, ", temp->data);
                temp = temp→next;    }        }
```

(a) Prints the contents of all the nodes
(b) Prints only the data of the head node
(c) Prints the contents of all the nodes except the last one
(d) Fails to print any data of the list

8. *n* random elements are stored in an array. Selection sort involves placing each element in correct position in the array. The time complexity of this process is going to be

(a) O(nlogn) **(b)** O(n) **(c) O(n^2)** **(d)** O(2^n)

9. Suppose that the complexity of an algorithm is $O(n^2)$. Suppose that the program that uses the algorithm run in 10 seconds for a data set of size n. If the data size is doubled, how long will it take (approximately) to run the program?

(a) 10 seconds **(b)** 100 seconds **(c)** 6-7 minutes **(d)** **None of the above**

10. Suppose there are two singly linked lists both of which intersect at some point and become a single linked list. The head or start pointers of both the lists are known, but the intersecting node and lengths of lists are not known. What is worst-case time complexity of optimal algorithm to find intersecting node from two intersecting linked lists?

(a) Θ (n * m), where m, n are lengths of given lists
(b) Θ (n^2), where m>n and m, n are lengths of given lists
(c) **Θ (n + m), where m, n are lengths of given lists**
(d) Θ (min(n, m)), where m, n are lengths of given lists

11. 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) **Delete the last element of the list**
(c) Insert a new element as a first element
(d) Add a new element at the end of the list

12. Consider an implementation of unsorted doubly linked list. Suppose it has its representation with a head pointer only. Given the representation, which of the following operation can be implemented in O(1) time?
(I) Insertion at the front of the linked list
(II) Insertion at the end of the linked list
(III) Deletion of the front node of the linked list
(IV) Deletion of the end node of the linked list

(a) I and II **(b)** **I and III** **(c)** I, II and III **(d)** I, II, III and IV

13. What would be the asymptotic time complexity to add a node at the end of singly linked list, if the pointer is initially pointing to the head of the list?

(a) O(1) **(b)** O(n) **(c)** **θ (n)** **(d)** θ (1)

14. Consider an implementation of unsorted doubly linked list. Suppose it has its representation with a head pointer and tail pointer. Given the representation, which of the following operation can be implemented in O(1) time?

(I) Insertion at the front of the linked list
(II) Insertion at the end of the linked list
(III) Deletion of the front node of the linked list
(IV) Deletion of the end node of the linked list

(a) I and II **(b)** I and III **(c)** I, II and III **(d)** **I, II, III and IV**

Ques.3 Give the tightest possible upper bound for the *worst case* running time for the following operations in terms of *N* or O(1) if constant time complexity. (4 Marks)

a. Given an unsorted array of size N, each insert operation will write a value into the array at the next available location. The index of this location is stored in the integer variable next_loc. next_loc is updated (incremented) after each insert. Once the array is full, the N+1st insertion will cause a new array of size 2N

to be created, and all N+1 values to be copied into the new array. What is the time complexity of an individual insert operation? **O(1)**

b. Printing out all the odd values stored in a linked list containing N positive integers *in ascending order*. **O(N)**

c. Determining all duplicates in an unsorted singly LL. **$O(N^2)$**

d. Determining all duplicates in a sorted singly LL. **O(N)**

Ques.4 Find the complexity of below code fragment: (1 Mark)

```
int happy (int n, int m)
{       if (n < 10)
                return n;
        else if (n < 100)
                return happy (n - 2, m);
        else
                return happy (n/2, m);          }
```

O(log n)

Ques.5 For each of the following scenarios, choose the "best" data structure from the following list or a combination of data structures: an unsorted array, linked list, DLL, circular LL, stack, queue. (5 Marks)

a. Suppose that a grocery store decided that customers who come first will be served first. **Queue**

b. A list must be maintained so that any element can be accessed randomly. **Array**

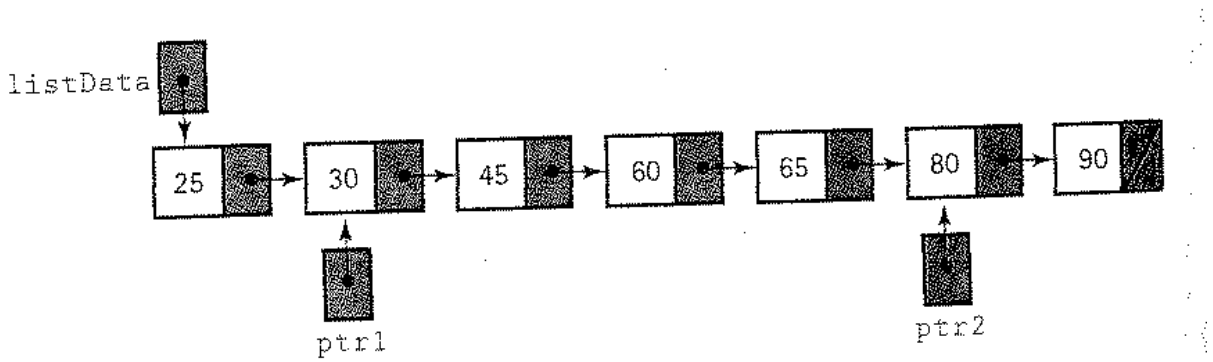
c. A program needs to remember operations it performed in opposite order. **Stack**

d. The size of a file is unknown. The entries need to be entered as they come in. Entries must be deleted when they are no longer needed. It is important that structure has flexible memory management. **Linked List**

e. A list must be maintained so that elements can be added to the beginning or end in O(1). **Circular LL**

Below Questions to be done in Answer Sheet:

Ques.6 Use the linked list pictured below to answer the questions from 1 to 4.



1. **Give the values for the following expressions:** (1.5 Marks)

(a) ptr1 -> info **30**
(b) ptr2 -> next -> info **90**
(c) listData -> next -> next -> info **45**

2. **Are the following expressions true or false?** **(2 Marks)**

(a) listData -> next = = ptr1 **True**
(b) ptr1 -> next -> info = = 60 **False**
(c) ptr2 -> next = = NULL **False**
(d) listData -> info = = 25 **True**

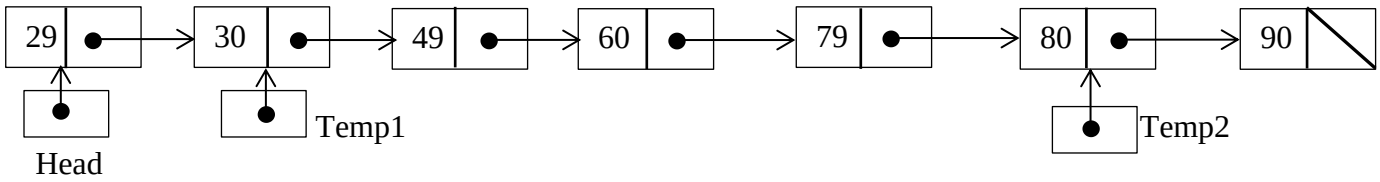
3. **Decide whether each of the following statements is valid or invalid. If it is valid, mark it OK; if it is invalid, explain what is wrong.** **(5 Marks)**

(a) listData -> next = ptr1 -> next; **Valid**
(b) listData -> next = *(ptr2 -> next); **Invalid**
(c) *listData = ptr2; **Invalid**
(d) ptr2 = ptr1 -> next -> info; **Invalid**
(e) ptr1 -> info = ptr2 -> info **Valid**

4. **Write one statement to do each of the following:** **(1.5 Marks)**

(a) Make listData point to the node containing 45. **listData = ptr1->next**
(b) Make ptr2 point to the last node in the list. **ptr2 = ptr2->next**
(c) Make listData point to an empty list. **listData = NULL**

Ques.7 Use the linked list pictured below to answer the questions 1 to 3. **(3 Marks)**



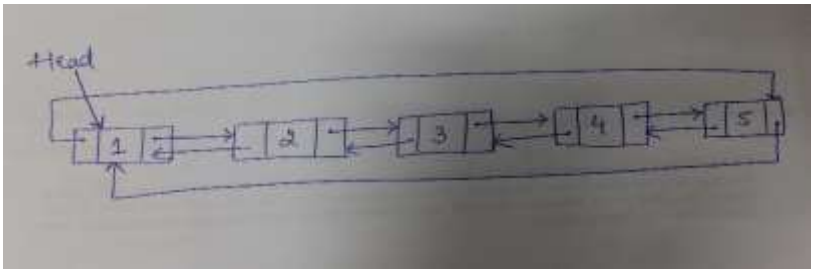
(1) What is the value of **Head -> next -> next -> data**? **49**
(2) Pick the right option from the following:
(a) Head -> next == Temp1
(b) Temp1 -> next -> data == 60
(c) Temp2 -> next -> next == NULL

(i) Only (a) is true.
(ii) Only (b) is true.
(iii) Both (a) and (b) are true.
(iv) Both (a) and (c) are true.
(v) Both (b) and (c) are true.

(3) Decide whether the syntax of each of the following statements is valid or invalid.
I. Head -> next = Temp1 -> next;
II. Head -> next = *(Temp2 -> next);
III. * Head = Temp2;

(a) **Only (I) is valid**
(b) Only (II) is valid
(c) Only (III) is valid
(d) Both (I) and (III) are valid
(e) Both (II) and (III) are valid.

Ques.8 Consider the circular doubly linked list given below: **(1*3 = 3 Marks)**



Consider the function given below:

```
struct node{
        struct node *lptr;
        int data;
        struct node *rptr;
};

void A( )
{
        struct node *t = head -> rptr;
        while(t != head)
        {
                swap(t -> lptr, t -> rptr);
                t = t -> lptr;
        }
        swap(head -> lptr, head -> rptr);
}
```

(a) When **'A'** function is called on the given list, the head right pointer (head->rptr) will point to ____5_____.

(b) Draw the new list formed after completion of **'A'** function. **1 -> 5 -> 4 -> 3 -> 2**

(c) To which node N (pointer) points when the following operation is performed on the new list formed.

struct node *N;
N = head -> rptr -> rptr -> lptr -> lptr -> rptr; **5**

Ques.9 Assume that LL is a DOUBLY linked list with the head node and at least one other internal node M which is not the last node. Write few lines of code to accomplish the following. You may assume that each node has a next pointer and prev pointer. You may NOT swap data to accomplish any of the following operations. For each operation, assume the original list as described above. You are encouraged to draw pictures to justify your code. Note that for each operation, you need to manipulate at least two pointers, next and prev. **(5 Marks)**

1. Delete the head node

```
head = head->next;
free(head->prev);
head->prev = NULL;
```

2. Insert a node P immediately after M

```
P->next = M->next;
P->prev = M;
```

```
M->next->prev = P;
M->next = P;
```

OR

```
P->next = M->next;
M->next = P;
P->next->prev = P;
P->prev = M;
```

3. Swap head and the node M (you may not swap data)

```
M->prev->next = head;
M->prev = head->prev;
head->prev = M->prev;
temp = M->next;
M->next = head->next;
head->next->prev = M;
head->next = temp;
temp->prev = head;
```

Total 7 pointers need to be changed

****