COEP TECHNOLOGICAL UNIVERSITY (COEP Tech)

A Unitary Public University of Government of Maharashtra
(Formerly College of Engineering Pune (COEP))

MID Semester Examination

Programme: B. Tech./~~B. Plan./M. Tech./M. Plan/MBA~~ **Semester: III**

Course Code: PCC - 04 **Course Name: Data Structures and Algorithms**

Branch: Computer Science and Engineering **Academic Year: AY 2024-25**

Duration: 1.5 hrs **Max Marks: 30**

Student PRN No.

6	1	2	2	0	3	0	1	2

Instructions:

1. Figures to the right indicate the full marks.
2. Mobile phones and programmable calculators are strictly prohibited.
3. Writing anything on question paper is not allowed.
4. Exchange/Sharing of stationery, calculator etc. not allowed.
5. Write your PRN Number on Question Paper.

			Marks
Q 1	✓	**Draw a pointer diagram for the below C code.**	3

```
typedef  struct node {
    int m;
    struct node *left, *right;
}node;
typedef node *tree;
void insert(tree *r, int i) {
    tree t, p, q;
    int j;
    t = (node *)malloc(sizeof(node));
    t->m = i;
    t->left = t->right = NULL;
    if(*r == NULL) {
        *r = t;
        return;
    }
    q = p = *r;
    j = i;
    while(i) {
        if(!p) q = p;
        if(i % 2 && p)
            p = p->left;
        else if(p)
            p = p->right;
        i--;
    }
    if(j % 2)
        q->left = t;
        else
            q->right = t;
        return;
}
int main() {
        int i;
        tree p;
        p = NULL;
        for(i = 0; i < 3; i++) {
            insert(&p, i);
        }
}
```

b **Consider the following magic function pseudo code that does some operation on an integer** — 1

```
void  magic (int n) {
        stack s;  // assume a empty stack created
        while (n > 0) {
            push(n%2, &s);
            n = n/2;
        }
        while (!empty(s))
            printf("%d", pop(&s)); // pop an element from s and
print it
}
```

What does magic(17) print?

c **What does this C function compute?** — 1

```
int xyz(char *c) {
    int sum = 0;
    while(*c) {
        if(*c >= '0' && *c <= '9')
            sum = sum * 10 + *c - '0';
        else
            break;
        c++;
    }
    return sum;
}
```

Q 2 a **Write following C functions with suitable prototypes for Array :** — 4

init() // initializes an array of 20 integers. **(0.5 mark)**

append( ) // to add an element in the array. **(1 mark)**

traverse() // to display all the elements of the array starting from first element to last. **(0.5 mark)**

mov_neg() /* This function moves all negative numbers to beginning and positive to end of the array. **(2 marks)**

For example,

		Input: -12 11 -13 -5 6 -7 5 -3 -6 **Output:** -12 -13 -5 -7 -3 -6 11 6 5 */	
	b	**Write following C functions with suitable prototypes for Array :** mov_freq() /*This function moves the most frequent element(s) to the start of the array, while maintaining the relative order of other elements. For example, **Input:** 4, 6, 4, 6, 7, 4 **Output:** 4, 4, 4, 6, 6, 7	2
	c	Given two polynomial numbers represented by a linked list. Write a C function to perform following operations: 1. structure for list. **(0.5 mark)** 2. List initialization. **(0.5 mark)** 3. Add element in list at end. **(1 mark)** 4. Perform Addition of two lists and store the result in third list. **(1 mark)** 5. Perform Subtraction of two lists and store the result in third list. **(1 mark)**	4
Q 3	a	**Write following C functions with suitable prototypes for ADT Circular Linked List :** **init_CLL()** // initializes circular linked list of integers. **(0.5 mark)** **insert_beg( )** // to add an element in the beg of the list. **(1 mark)** **traverse()** // to display all the elements of the list starting from first element to last. **(0.5 mark)** **del_odd()** /*This function deletes all odd integers in the list . **(1 mark)** For example, if the list is: {1, 2, 8, 9,12,16,18,11,14,13} after the function call the resultant circular list should be: {2, 8,12,16,18,14} */ You are free to include more functions if required.	3
	b	**Write following C functions with suitable prototypes for ADT Singly Linked List :** **init_SLL()** // **(0.5 Mark)** to initialize the list **insert_beg( )** // **(1 Mark)** to add an element at the beginning of the list. **traverse()** // **(0.5 Marks)** to display all the elements of the list starting from last element to first. **is_palindrome()** //**(1 marks)** /* The function determines the given list is a palindrome or not. For example, if the list is: {1, 2, 3, 2, 1} is a palindrome. For example, if the list is: {1, 2, 3, 1, 2, 1} is a not a palindrome.*/ You are free to include more functions if required.	3
	c	Write a C function to count the number of elements in a doubly linked list. Note: Write only count() function. Include structure of doubly linked list.	2

Q4 **a** An array of size 9 is given. Three stacks are implemented using this array in order 1,2 and 3. Size of each stack is : 1

Stack 1 - size 2
Stack 2 - size 3
Stack 3 - size 4

Initially array contains no elements.
Operation push(1,2) means push value 2 in stack number 1.
Assume all push operations happen from right of stack. After performing the following operations, what are the contents of the array starting from index 0?

```
push(1,2)
push(2,3)
push(3,4)
push(1,5)
push(2, pop(1))
push(2,6)
push(3,8)
push(3,7)
push(3,pop(1))
push(1,pop(2))
push(1,5)
push(2,1)
```

b **Write following C functions with suitable prototypes for ADT Stack using array :** 3

init_stack() // Initializes the stack with the given capacity. **(0.5 mark)**
push( ) // Adds an element to the stack. **(0.5 mark)**
pop() // Removes the top element from the stack **(0.5 mark)**
sortStack() // Sorts the original stack in ascending order using only one additional stack by comparing and transferring elements between the original and temporary stack. **(1 mark)**
Example: Input: Stack [3, 5, 1, 4, 2] Output: Stack [1, 2, 3, 4, 5]
printStack() // Displays the stack contents. **(0.5 mark)**
You are free to include more functions if required.

c Convert the following infix expression in its postfix form using stack. Show step wise conversion. 3

A + B * (C ^ D - E) ^ (F + G * H) – I