COEP TECHNOLOGICAL UNIVERSITY (COEP Tech)

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

END 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: 2024-25**

Duration: 3 hrs **Max Marks: 60**

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	CO	PO
Q 1	a	In order to concatenate two linked list in O(1) time, which variation of linked list can be used?	1	2,5	1
	b	What is the worst-case time complexity of inserting a node in a sorted doubly linked list?	1	5	1,2
	c	In linked list implementation of queue, if only front pointer is maintained, which of the queue operation(s) would take worst case linear time?	1	1,2	3

Q 2	a	`#include<stdio.h>`
		
		`typedef struct data{`
		`  char x,y,z;`
		`}data;`
		`int main(){`
		`  data p = {'1', '0', 'a'+2};`
		`  data *q = &p;`
		`  printf ("%c, %c", *((char*)q+1), *((char*)q+2));`
		`  return 0;`
		`}`
		Write the output of above code.
			2	4	1,2

	b	What is the output of the code given below? Assume the standard definition of stack functions.
		`#include <stdio.h>`
		`#include "stack.h"`
		`/* assume that stack doesn't get full */`
		`int main() {`
		`  stack s;`
		`  int i;`
		`  init(&s);`
		`  for(i = 0; i < 8; i++)`
			2	4	2

P.T.O

Scanned with OKEN Scanner

```
        if(i % 3 == 0)
            push(&s, i + 1);
        if(i % 5 == 0)
            push(&s, i - 1);
    while(!isempty(&s))
        printf("%d\n", pop(&s));
    return 0;
}
```

	c	Consider an unsorted array A = {5, 1, 4, 2, 8}. Sorting of elements is done using bubble sort. How many passes will it require to get the array sorted. Note: Write only numeric values as answer.	2	5	3
Q 3	a	You are given a mathematical expression in **postfix notation**. Implement `evaluate_postfix()` function which evaluates the postfix expression using a **stack** data structure and returns an integer result. Assume the required stack operations. The function prototype is given below. `int evaluate_postfix(char* expression);` Note : Operands and operators are separated by one space.	3	3	3
	b	Write following functions in C with suitable prototypes for ADT Circular Singly Linked List : `init_SLL()` **(0.5 marks)** // initializes circular singly linked list of integers `append( )` **(1 mark)** // to add an element in the end of the list. `traverse()` **(0.5 mark)** // to display all the elements of the list starting from first element to last. `remove_duplicates()` **(2 marks)** /* The functions removes a duplicate node keeping only one node so that elements of node are unique. For example, if the list is: {1, 2, 3, 2, 1} after the function is invoked the list changes to: {1, 2, 3 } */ You are free to include more functions if required .	4	2,3	2,3
	c	Finishing the interiors of a lecture hall consists of several steps, such as laying electrical cables, installing audio-visual equipment, attaching the blackboard, etc. Suppose the steps are named as A, B, C, D, F and G. There is a directed edge from step 'X' to step 'Y' iff 'X' has to completed before 'Y'. Below is the order in which work can proceeded: A must happen before B B must happen before C C must happen before D D must happen before G A must happen before E E must happen before F E must happen before D D must happen before F F must happen before G What is the minimum number of days required to complete the interiors, if step takes a day to complete and independent steps can be completed parallelly?	2	5	2

Scanned with OKEN Scanner

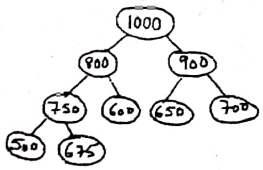
Q 4	a	You are given an ADT stack with **push** and **pop** operations. Implement an ADT **queue** using two instances of stack data structures and their operations. Let the queue be represented by q and the stacks used to implement the queue be stack1 and stack2. Implement the following operations for the queue: 1. `void init_queue(Queue* q);` // Initializes the queue q by creating two empty stacks stack1 and stack2. (0.5 mark) 2. `void enqueue(Queue* q, int data);` // Adds an element to the end of the queue using stack1. (1 mark) 3. `int dequeue(Queue* q);` // Removes the element from the front of the queue (1 mark) 4. `void display_queue(Queue* q);` // Displays the elements of the queue. (0.5 mark) Include the structure for both Queue and Stack in your solution.	3	3,4	3
	b	Design and implement a Heap ADT in C to perform Heap Sort on an array of integers. **Function Specifications:** • `Heap* CreateHeap(int maxSize)` //Initializes a new heap structure with the specified maximum size. Returns a pointer to the created heap. • `void Insert(Heap* heap, int value).` //Inserts the given value into the heap while maintaining the max-heap property. • `void Heapify(Heap* heap, int* arr, int n)` //Converts the given array of size n into a max-heap and stores it in the heap. • `void HeapSort(int* arr, int n)` //Sorts the array in ascending order using the heap.	4	3	3
	c	You are given two polynomials represented by linked lists, where each node in the list represents a term in the polynomial with its coefficient and power of the variable. Write a C function to add these two polynomials by combining the terms with the same power and return the resultant polynomial in the form of a linked list. Implement the following operations for the Polynomial ADT: 1. Define the **structure for the polynomial list node**: Each node should store the coefficient, the power of the variable, and pointers to the next node. **(0.5 marks)** 2. `void init_list(Polynomial *p);` //Initializes an empty linked list for the polynomial. **(0.5 marks)** 3. `void add_term(Polynomial *p, int coefficient, int power);` //Adds a term with the specified coefficient and power at the end of the polynomial list. **(1 mark)** 4. `Polynomial add_polynomials(Polynomial *p1, Polynomial *p2);` //Performs the addition of two polynomial lists, p1 and p2, by adding the coefficients of terms with the same power. Stores the result in a new linked list and returns it. **(1 mark)** 5. `void display_polynomial(Polynomial p);` //Displays the polynomial in the standard format (e.g., 5x^2 + 3x + 2). **(1 mark)**	4	2,3	3
Q 5	a	Implement an ADT for a queue using a doubly circular linked list. The queue should maintain both a head pointer (to the first element) and a tail pointer (to the last element) of the sequence. Implement the following functions. Write your structure for	5	4,5	3

Scanned with OKEN Scanner

Queue.

1. `void init_queue(Queue* q);` //Initializes an empty queue.
2. `void enqueue(Queue* q, int data);` //Adds an element with the specified data to the end of the queue. Updates both the head and tail pointers as necessary.
3. `int dequeue(Queue* q);` //Removes the element from the front of the queue and returns its data. Updates the head and tail pointers appropriately.
4. `int is_empty(Queue* q);` //Checks if the queue is empty. Returns 1 if it is empty, and 0 otherwise.
5. `void display_queue(Queue* q);` //Displays all elements in the queue from front to back.
6. `int is_full(Queue* q);` //A trivial function that returns 0, assuming the queue is never full.

	b	Write a recursive C function to verify whether a given binary tree is a Binary Search Tree (BST). (Mention the tree implementation used i.e., Array or Pointer). The Binary Tree structure is given below.
		
		`typedef struct {`
		`  Node* root;  // Pointer to the root node of the binary tree`
		`} BT;`
		
		Implement the following function:
		`int is_bst(BT* t);`
		
		The above function Returns 1 if the tree is a Binary Search Tree, and 0 otherwise. The function should use recursion to validate that the tree meets the conditions of a Binary Search Tree. Consider all nodes and subtrees to ensure the global ordering property holds.
			5	3,4	3

	c	A tree of nodes having block size of memory in KB is available as shown in an example in the diagram below:  A request for **m KB** (say for example 580 KB) of dynamic memory is received by OS. Write a function **findBlock()** to return the block size which best fits the requested size of memory from the tree in this case 600KB. Note: Parameters to the function are tree and block size.	5	3,4	3
Q6	a	Design an ADT for a city's bus transportation network using a graph represented by an adjacency list. Each bus stop is a node, and each route between stops has a specific distance in kilometres. Implement the following functions:	8	3,4	3

Scanned with OKEN Scanner

1. `InitializeGraph(Graph *g, int numOfStops)` //Initializes the graph with the specified number of stops. **[0.5 Mark]**
2. `AddRoute(Graph *g , int stop1, int stop2, int distance)` //Adds a route between two stops with the given distance. **[2 Marks]**
3. `RemoveRoute(Graph *g, int stop1, int stop2)` //Removes an existing route between two stops. **[1 Mark]**
4. `GetConnectedStops(Graph *g, int stop)` //Returns a list of stops directly connected to the given stop. **[2 Marks]**
5. `ShortestPath(Graph *g, int startStop, int endStop)` //Finds the shortest path between two stops using Dijkstra's algorithm. **[2 Marks]**
6. `DisplayNetwork(Graph g)` //Prints the entire network, showing each stop and its connections with distances. **[0.5 Mark]**

The structure for the Graph is given below. Use the same.

```
// Structure for each node in the adjacency list
typedef struct AdjNode {
    int stop;                   // Bus stop identifier
    int distance;               // Distance to the connected
stop
    struct AdjNode* next;       // Pointer to the next
connected stop
} AdjNode;

// Structure for the graph
typedef struct Graph {
    int numStops;               // Total number of bus stops
    AdjNode** adjList;          // Array of adjacency lists
for each stop
} Graph;
```

b	Implement a C function that detects whether a cycle exists in an undirected graph. The graph is represented by an ADT which allows adding edges and performing a Depth-First Search (DFS) for cycle detection. Your task is to implement a function `has_cycle()` that returns 1 if the graph contains a cycle, and 0 if it does not. Use the below definition of graph and adjacency list structure	4	4.5	1.2

```
// Definition of the graph structure
typedef struct {
    int V; // Number of vertices
    struct Node** adjList; // Adjacency list
}Graph;

// Structure for adjacency list node
typedef struct Node {
    int vertex;
    struct Node* next;
}Node;

// Function to detect cycle in the graph using DFS
int has_cycle(Graph* graph);
```

Scanned with OKEN Scanner

Comment on the time complexity of the implemented function to detect cycle.

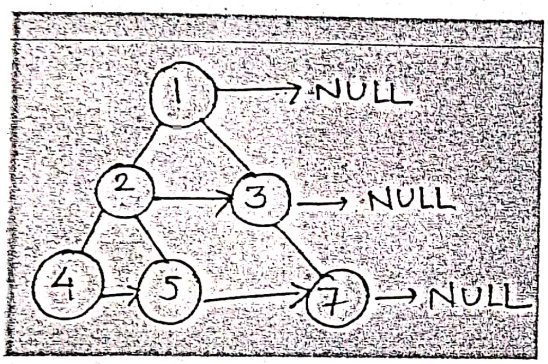
c Consider the Binary Tree having structure as below: 4 3,4 3

```
typedef struct mynode {
  int val;
  Node *left;
  Node *right;
  Node *next;
}mynode
```

Write a recursive or non-recursive function : `mynode* connect(mynode* root);`

The function `connect()` should set next pointer to point to its next right node. If there is no next right node, the next pointer should be set to NULL.

Example:



********** ALL THE BEST **********

Scanned with OKEN Scanner