SET C- ANSWER KEY

SRM Institute of Science and Technology
College of Engineering and Technology
School of Computing
SRM Nagar, Kattankulathur – 603203, Chengalpattu District, Tamilnadu
Academic Year: 2023-24 (ODD)

Test: CLA-CT-1 **Date: 04-09-2023**
Course Code & Title: 21CSC201J Data Structures and Algorithms **Duration:** 1 hour 40 min
Year & Sem: II Year / III Sem **Max. Marks:** 50

Course Articulation Matrix:

Course Outcome	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	Program Specific Outcomes		
													PSO-1	PSO-2	PSO-3
CO1	1	2	3	-	-	-	-	-	-	-	-	3	3	-	-
CO2	2	3	3	-	-	-	-	-	-	-	-	3	3	-	-
CO3	2	3	3	-	-	-	-	-	-	-	-	3	3	-	-
CO4	2	3	3	-	-	-	-	-	-	-	-	3	3	-	-
CO5	3	2	3	-	-	-	-	-	-	-	-	3	3	-	-

Part - A(10 x 1 = 10 Marks) **Instructions: Answer all**

Q. No	Question	Marks	BL	CO	PO	PI Code
1	In a linear search algorithm, the worst-case scenario occurs when ……. A. Item is somewhere in the middle of the array B. Item is not in the array at all C. Item is the last element in the array D. Item is the last element in the array or item is not there at all	1	L1	1	2	2.5.1
2	Two main measures of the efficiency of an algorithm are a) Processor and memory b) Complexity and capacity c) Time and space complexity d) Data and space	1	L1	1	1	1.7.1

3	What will be the output for the following C code?
	#include<stdio.h>
	void main() {
	int a[] = {5,7,3,2,4}, *p;
	p = a;
	++*p;
	printf("%d ", *p);
	p += 3;
	printf("%d ", *p);
	}
	a)7 4
	b)9 4
	c)6 2
	d)2 3
		1	L2	1	1	1.7.1

4	If ‘p’ is declared as integer pointer, then, an array ‘a’ can be pointed by the following assignment – a) p = &a[0] b) p++ c) p == &a[0] d) p == a[0]	1	L2	1	2	2.5.2
5	Which function gives the positive value of the given input? a) Absolute b) Logarithmic c) Floor d) Ceiling	1	L1	1	2	2.5.1

6	Find out the correct option when adding a new node into the end of list a. Head is made to point to the new node, New node points to the previously first element. b. Last node now points to the new node, New node points to NULL. c. Previous node now points to the new node, New node points to the next node. d. Last node now points to the new node, New node points to the previously first element.	**1**	**L1**	**2**	**2**	**2.5.1**
7	Linked list is considered as an example of _____ type of memory allocation. a) Dynamic b) Static c) Compile time d) Heap	**1**	**L1**	**2**	**2**	**2.5.3**
8	Which type of linked list stores the address of the head node in the next pointer of the last node? a) Singly Linked List b) Doubly Linked List c) Hashed List d) Circular Linked List	**1**	**L1**	**2**	**2**	**2.5.1**

9	Which of the following is the correct declaration of self-referential structure?
	a) struct node*
	{
	int data;
	node *link;
	}
	
	b) struct node
	{
	int data;
	struct node *link;
	};
	
	c) struct node
	{
	int data;
	node *link;
	}
	
	d) struct node*
	{
	int data;
	struct node *link;
	}
		1	**L2**	**2**	**1**	**1.7.1**

10	Inserting a node in the middle of the singly linked list needs to modify _____ pointers. a) 1 b) 3 c) 2 d) 0	**1**	**L2**	**2**	**1**	**1.7.1**

SET C – ANSWER KEY

SRM Institute of Science and Technology
College of Engineering and Technology
School of Computing
SRM Nagar, Kattankulathur – 603203, Chengalpattu District, Tamilnadu
Academic Year: 2023-24 (ODD)

Test: CLA-CT-1 **Date: 04-09-2023**
Course Code & Title: 21CSC201J Data Structures and Algorithms **Duration:** 1 hour 40 min
Year & Sem: II Year / III Sem **Max. Marks:** 50

Course Articulation Matrix:

Course Outcome	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	Program Specific Outcomes		
													PSO-1	PSO-2	PSO-3
CO1	1	2	3	-	-	-	-	-	-	-	-	3	3	-	-
CO2	2	3	3	-	-	-	-	-	-	-	-	3	3	-	-
CO3	2	3	3	-	-	-	-	-	-	-	-	3	3	-	-
CO4	2	3	3	-	-	-	-	-	-	-	-	3	3	-	-
CO5	3	2	3	-	-	-	-	-	-	-	-	3	3	-	-

Part – B
(4 x 5 = 20 Marks)
Instructions: Answer ALL

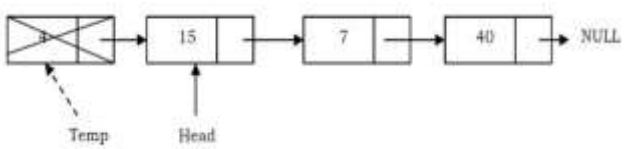
Q. No	Question	Marks	BL	CO	PO	PI Code

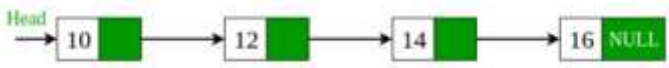
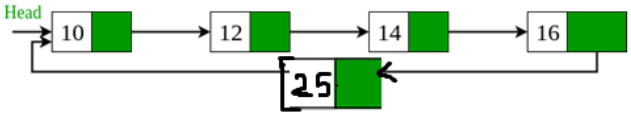
11	Create a structure named myCar with the array of structure members and with the following member variables: make, model and year.
	Answer:
	#define MAX_LEN (25)
	typedef struct _TCar
	{
	char make[MAX_LEN];
	char model[MAX_LEN];
	int year;
	} *TCar;
	
	void Car_Init (TCar car, char * Make, char * Model, int yr)
	{
	strcpy(car->make,Make);
	strcpy(car->model,Model);
	car->year = yr;
	}
	
	void Car_Output(TCar car, char * debugMsg)
	{
	
	printf(" Car make is >%s< \n",car->make);
	printf(" Car model is >%s< \n",car->model);
	printf(" Car year is %d \n",car->year);
		5	L3	1	1	1.7.1

	}
	
	int main()
	{
	struct _TCar myCar;
	Car_Init(&myCar,(char *) "Toyota",(char *) "Yaris",2012);
	Car_Output(&myCar, (char *) " my car");
	}					

12	Ramu runs a college that has 5 departments. He needs the details of students having highest CGPA between 9- 9.5. Help him to identify the students from various departments.
	Mandatory condition: create "struct student"
	
	#include <stdio.h>
	#include <string.h>
	
	struct student {
	char name[50];
	int department;
	float cgpa;
	};
	
	int main() {
	struct student students[100]; // Assuming you have up to 100 students
	int numStudents;
	
	printf("Enter the number of students: ");
	scanf("%d", &numStudents);
	
	// Input student details
	for (int i = 0; i < numStudents; i++) {
	printf("Enter details for student %d:\n", i + 1);
	printf("Name: ");
	scanf("%s", students[i].name);
	printf("Department (1-5): ");
	scanf("%d", &students[i].department);
	printf("CGPA: ");
	scanf("%f", &students[i].cgpa);
	}
	
	printf("Students with CGPA between 9 and 9.5:\n");
	
	// Filter and print students with CGPA between 9 and 9.5
	for (int i = 0; i < numStudents; i++) {
	if (students[i].cgpa >= 9.0 && students[i].cgpa <= 9.5) {
	printf("Name: %s\n", students[i].name);
	printf("Department: %d\n", students[i].department);
		5	**L3**	**1**	**1**	**1.7.1**

	printf("CGPA: %.2f\n", students[i].cgpa);
	}
	}
	
	return 0;
	}					

13	Write the pseudocode for deleting the node named 'Temp'  Point head to the next node i.e. second node temp = head head = head->next Make sure to free unused memory free(temp); or delete temp;	**5**	**L3**	**2**	**2**	**2.5.2**

14	Write an algorithmic routine to convert singly linked list into a circular linked list after inserting the data element 25 after the last node.
	 
	
	Answer:
	 
	
	struct Node* convertToCircular(struct Node* head) {
	if (head == NULL) {
	return NULL;
	}
	
	// Traverse to find the last node
	struct Node* current = head;
	while (current->next != NULL) {
	current = current->next;
	}
	
	// Create a new node with data 25
	struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));
	new_node->data = 25;
		5	**L3**	**2**	**2**	**2.5.2**

	new_node->next = NULL; **// Connect the last node to the new node** **current->next = new_node;** **// Connect the new node to the head** **new_node->next = head;** **// Update the head pointer** **head = new_node;** return head; }					
Part – C **(2 x 10 = 20 Marks)**						
15	a) Prove that n = O(nlogn) – **5 mark** To prove that n=O(nlogn), we need to show that there exists a constant c>0 and a value of n0 such that for all n≥n0, the inequality n≤c·nlogn holds true. Let's attempt to prove this by finding suitable values for c and n0. We want to show that n≤c·nlogn. To simplify, we can cancel out the common factor of n on both sides of the inequality: 1≤c·logn Now, we can isolate c by dividing both sides of the inequality by logn (assuming n>1 since logn is undefined for n≤1): c≥1/logn Now, let's choose c=2 (you can choose any constant c≥1 for the proof). So, we have c=2. Now, we need to find n0 such that for all n≥n0, 2≥1/logn. Let's choose n0=2 (you can choose any value greater than or equal to 2). Now, we have n0=2. For all n≥2, we can see that 2≥1/logn holds true because as n grows, logn grows slowly, and the reciprocal of logn decreases. Therefore, for n≥2, 2≥1/logn, and consequently, n≤2·nlogn. So, we have found a constant c=2 and a value n0=2 such that for all n≥n0, n≤2·nlogn, which proves that n=O(nlogn). b) Prove that $7n \neq \Omega(n2)$ **– 5 marks** we need to show that for any constant c>0 and sufficiently large n, 7n is not greater than or equal to	**10**	**L3**	**1**	**1**	**1.7.1**

	c·n2 Let's attempt to prove this by contradiction. Assume that 7n≥c·n2 for some constant c and for all sufficiently large values of n. Now, we can simplify this inequality: 7n≥c·n2 Dividing both sides of the inequality by n (since n is positive for sufficiently large n): 7≥c·n Now, it's clear that as n approaches infinity, the right side of the inequality (c·n) also approaches infinity. However, the left side of the inequality (7) is just a constant. This means that no matter how large you make the constant c, there will always be some sufficiently large value of n for which 7 is not greater than or equal to c·n. Therefore, we cannot find a constant c and a value of n such that 7n≥c·n2 holds true for all n greater than or equal to that value. Hence, we have proven by contradiction that 7n is not in the set Ω(n2).					
	(or)					

16	Write a pseudocode to define a structure for a hotel that has members - name, address, grade, number of rooms, and room charges. Write a function to print the names of hotels in a particular grade.
	
	Answer:
	
	struct Hotel {
	  char name[100];
	  char address[200];
	  int grade;
	  int numRooms;
	  float roomCharges;
	};
	
	// Function to print the names of hotels in a particular grade
	void printHotelsInGrade(struct Hotel hotels[], int numHotels, int targetGrade) {
	  for (int i = 0; i < numHotels; i++) {
	  if (hotels[i].grade == targetGrade) {
	  printf("Hotel Name: %s\n", hotels[i].name);
	  }
		10	**L3**	**1**	**1**	**1.7.1**

	}
	}
	
	int main() {
	// Example hotels
	struct Hotel hotels[3];
	
	// Fill in hotel data (for demonstration purposes)
	strcpy(hotels[0].name, "Hotel A");
	strcpy(hotels[0].address, "123 Main St");
	hotels[0].grade = 3;
	hotels[0].numRooms = 50;
	hotels[0].roomCharges = 100.0;
	
	strcpy(hotels[1].name, "Hotel B");
	strcpy(hotels[1].address, "456 Elm St");
	hotels[1].grade = 4;
	hotels[1].numRooms = 75;
	hotels[1].roomCharges = 150.0;
	
	int targetGrade = 4;
	printf("Hotels in Grade %d:\n", targetGrade);
	printHotelsInGrade(hotels, 3, targetGrade);
	return 0;
	}					

17	In a warehouse, boxes are arranged in places where their size is fixed. Mr.Tilak, the store in- charge, wants to insert the box in the middle and he visits all the boxes one by one from start to end. Write the appropriate pseudo code for the above-mentioned scenario.
	
	Routine for Insertion:
	int LA[] = {1,3,5,7,8};
	int item = 10, k = 3, n = 5;
	int i = 0, j = n;
	for(i = 0; i<n; i++) {
	print("LA[%d] = %d \n", i, LA[i]);
	}
	n = n + 1;
	while(j >= k) {
	LA[j+1] = LA[j];
	j = j - 1;
	}
	LA[k] = item;
	for(i = 0; i<n; i++) {
	print("LA[%d] = %d \n", i, LA[i]);
	}
	}
		10	**L3**	**2**	**2**	**2.5.2**

	(or)					
18	Ramu's mothers is asking him to buy the list of things (Item A, Item B, Item C, Item D, Item E and Item F) needed for their home. She gave the list to	**10**	**L3**	**2**	**2**	**2.5.2**

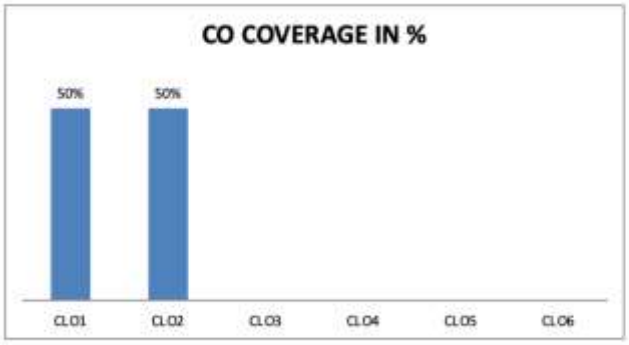
Ramu to buy the things in the market. After receiving the list, Ramu went to the market to buy the things. Ramu's mother came to know that Item A is already in the home. So immediately she contacted Ramu and told him to remove Item **A** from the list and add **Item G** in that place. Implement this concept using a circular doubly linked list to prepare the new list after removing item A and adding item G.

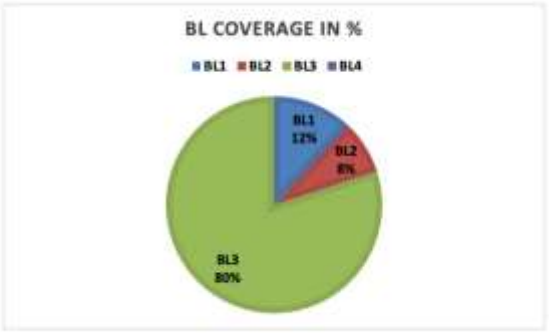
```
// Define the structure of a node
struct Node {
    char data;
    struct Node* next;
    struct Node* prev;
};

// Function to insert a new node at the end of the circular doubly linked list
struct Node* insertAtEnd(struct Node* head, char data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;

    if (head == NULL) {
        head = newNode;
        head->next = head;
        head->prev = head;
    } else {
        struct Node* last = head->prev;

        last->next = newNode;
        newNode->prev = last;

        newNode->next = head;
        head->prev = newNode;
    }

    return head;
}

// Function to remove a node with a specific data value from the circular doubly linked list
struct Node* removeNode(struct Node* head, char data) {
    if (head == NULL) {
        return NULL;
    }

    struct Node* current = head;
    while (current->data != data) {
        current = current->next;
        if (current == head) {
            printf("Item %c not found in the list.\n",
```

	data);
	return head;
	}
	}
	
	struct Node* prevNode = current->prev;
	struct Node* nextNode = current->next;
	
	prevNode->next = nextNode;
	nextNode->prev = prevNode;
	
	if (current == head) {
	head = nextNode;
	}
	free(current);
	return head;
	}					

***Program Indicators are available separately for Computer Science and Engineering in AICTE examination reforms policy.**

Course Outcome (CO) and Bloom's level (BL) Coverage in Questions





Approved by the Audit Professor/Course Coordinator