Sardar Patel Institute of Technology

Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai-400058, India
(Autonomous College Affiliated to University of Mumbai)

Mid Semester Examination

October 2023

Max. Marks: 30 Duration: 60 mins.

Class: SE Semester: III
Course code: CS202 Branch: CS/DS/AIML
Name of the Course: Data Structures

Instruction:
(1) All questions are compulsory
(2) Draw neat diagrams to
(3) Assume suitable data if necessary

Q. No.		Max. Mks	CO-BL-PI
1 a	i) Explain the concept of **stack overflow** and how it can be avoided in a stack implementation.	01	1-3-1.3.1
	ii) Think of an algorithm that uses a Stack to efficiently check for **unbalanced brackets**. What is the maximum number of characters that will appear on the stack at any time when the algorithm analyzes the string "([]O[O])"? Show your reasoning with the states of the stack as the algorithm parses the string.	02	1-3-1.4.1

1 b	Implement a **Queue** ADT that uses 2 stacks instead of an array as the underlying storage structure.
	Complete the following function for this ADT in C:
	`char dequeue(struct Queue* queue) {`
	`}`		

	i) ADT structure ii) dequeue implementation You have access to the following structure and functions: `typedef struct Stack{ int top; int size; char* array; } Stack;` → bool **is_stack_empty**(Stack * s); → bool **is_queue_empty**(Queue * q); → char **pop**(Stack* stack); → void **push**(Stack *stack, char item)	01 02	1-3-1.4.1

1c	i. What is an ADT? List one non-linear data structure along with its application.
	
	ii. Write a C function:
	`int evaluate_prefix(char *expression);`
	to evaluate a prefix expression using a stack. The following structure & functions are available to you
	
	`typedef struct Stack{`
	`  int top;`
	`  int size;`
	`  char* array;`
	`} Stack;`
	
	`Stack* init_stack(int size);`
	
	`void push(Stack *stack, char item);`
	
	`char pop(Stack* stack);`
	
	`// returns 1 if ch is an operator`
	`int is_operator(char ch);`
	
	`// returns result of operating opnd1 & opnd2 with 'op'`
	`int perform_operation(int opnd1, int opnd2, char op)`
	
	iii. In the **Josephus problem**, a group of people stand in a circle. Starting at the 1st position, every k-th person is eliminated from the circle, and the process continues until only one person remains. If there are **41** people initially, and every 2nd person is eliminated, find the final survivors after round 4 of eliminations. Show queue states for all rounds.
	
	OR
	
	i. State any **2** conditions that need to be satisfied for implementing **recursion** using stacks.
	
	ii. Write a function to **reverse the contents of a queue** using an **aux stack**. The following structures & functions are available to you:
	
	`typedef struct Stack{`
	`  int top;`
	`  int size;`
	`  char* array;`
	`} Stack;`
	
	`Stack* init_stack(int size);`
	
	`void push(Stack *stack, char item)`
	
	`char pop(Stack* stack);`
	
	`typedef struct Queue{`
	`  int front, rear;`
	`  int size;`
	`  char* array;`
	`} Queue;`
	
	`char dequeue(Queue* queue);`
	
	`void enqueue(Queue *queue, char item)`
		1
		
		1.5
		
		1.5
		
		1
		
		1.5
			1-3-1.3.1
			
			1-3-1.4.1
			
			1-3-2.1.2
			
			1-3-1.3.1
			
			1-3-1.4.1

	bool **is_stack_empty**(Stack * s); | bool **is_queue_empty**(Queue * q); iii. What is the optimal number of moves needed to solve the **Towers of Hanoi** problem with **10 disks?** Draw the disks configuration after the **511th** optimal move.	1.5	1-3-2.1.3

2 a	The following function takes a singly-linked list of integers as a parameter and rearranges the elements of the list. The function is called with the list containing the integers 1, 2, 3, 4, 5, 6, 7 in the given order. What will be the contents of the list after the function completes execution? Justify your answer for the given list elements.
	
	class Node {
	int value;
	Node next;
	}
	void rearrange(Node list)
	{
	Node p, q;
	int temp;
	if (list == null || list.next == null)
	{
	return;
	}
	p = list;
	q = list.next;
	while (q != null)
	{
	temp = p.value;
	p.value = q.value;
	q.value = temp;
	p = q.next;
	q = p != null ? p.next : null;
	}
	}
	OR
	
	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 parts of the code are left blank. Complete the code given below by filling the blank lines.
		3	1-3-1.4.1

	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;
	}		

2 b	Write down the structure of a node in the linked list where node holds a float value. Why do you require a linked list when an array is available with you?	01 03	1-3-1.3.1
2 c	Given a singly linked list of characters, write a function that returns true if the given list is a palindrome, else false.	03	1-3-1.4.1
3 a	Construct a Binary Search Tree by inserting the data in the given order: All steps to be shown. 21, 18, 33, 35, 20, 13, 41, 55. Traverse the constructed tree using all traversing techniques.	03 03	2-3-4.1.2
3b	Which data structures can be used for construction of an expression tree? Construct an expression tree for the given postfix expression using the same data structure: Clearly show each step in construction. a b + c d e + * *	01 03	2-3-4.1.2