MANIPAL INSTITUTE OF TECHNOLOGY

MANIPAL

(A constituent unit of MAHE, Manipal)

III SEMESTER B.TECH. END SEMESTER EXAMINATION

DECEMBER 2023

SUBJECT: DATA STRUCTURES AND ALGORITHMS [MTE 2121]

Date of Exam: **9/12/2022** Time of Exam: **02:00 PM – 05:00 PM** Max. Marks: **50**

Instructions to Candidates:

- Answer ALL the questions & missing data may be suitably assumed

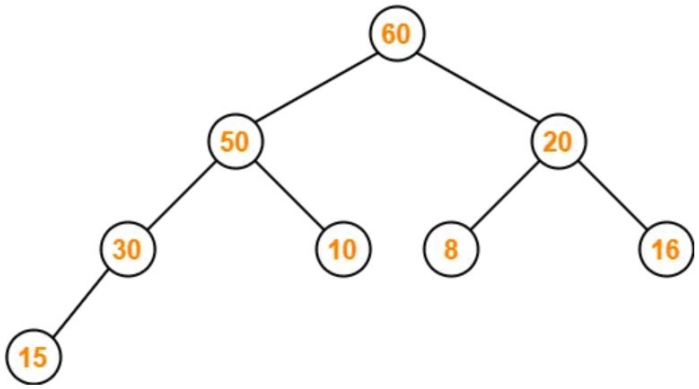
Q.NO.	QUESTION	M	CO	PO	LO	BL
1.	Appraise the limitations of the Array data structure and the need for dynamic array resizing. Use a suitable example to support your answer.	2	1	1, 2, 3, 4, 5, 12	1, 2	4

2.	Given below is a code snippet for the creation of a Queue class.
	`# Completed implementation of a queue ADT`
	`class Queue:`
	`  def __init__(self):`
	`  self.items = []`
	`  def is_empty(self):`
	`  return self.items == []`
	`  def enqueue(self, item):`
	`  self.items.insert(0,item)`
	`  def dequeue(self):`
	`  return self.items.pop()`
	`  def size(self):`
	`  return len(self.items)`
	Determine the output of the program after the following statements are executed.
	q = Queue()
	q. enqueue ('End')
	q. enqueue ('Semester')
	q. enqueue ('Examination')
	q. dequeue ()
	q. is_empty()
	q. dequeue ()
	q. size()
		3	1	1, 2, 3, 4, 5, 12	1, 2	3

3.	Using the given cheat sheet, develop a Python program for a dictionary to store the values of Employee ids and Employee names of an organization. Devise the commands to showcase, the key-value pairs in the dictionary. Finally, showcase the usage of the get ( ) function to retrieve a null key-value pair from your dictionary. **Operator** / **Use** / **Explanation** `[]` / `my_dict[k]` / Returns the value associated with k, otherwise its an error **`in`** / `key in my_dict` / Returns True if key is in the dictionary, False otherwise **`del`** / `del my_dict[key]` / Removes the entry from the dictionary **Method Name** / **Use** / **Explanation** `keys` / `my_dict.keys()` / Returns the keys of the dictionary in a `dict_keys` object `values` / `my_dict.values()` / Returns the values of the dictionary in a `dict_values` object `items` / `my_dict.items()` / Returns the key-value pairs in a `dict_items` object `get` / `my_dict.get(k)` / Returns the value associated with k, `None` otherwise `get` / `my_dict.get(k,alt)` / Returns the value associated with k, alt otherwise	**5**	**2**	**1, 2, 3, 4, 5, 9, 12**	**1, 2, 12, 16, 17**	**6**
4.	Create a function in Python find the sum of all numbers from 1 to 10 using the principle of recursion.	**2**	**2**	**1, 2, 3, 4, 5, 9, 12**	**1, 2, 12, 16, 17**	**6**

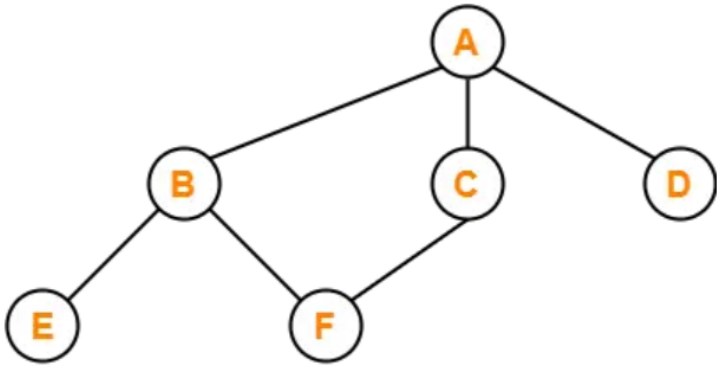
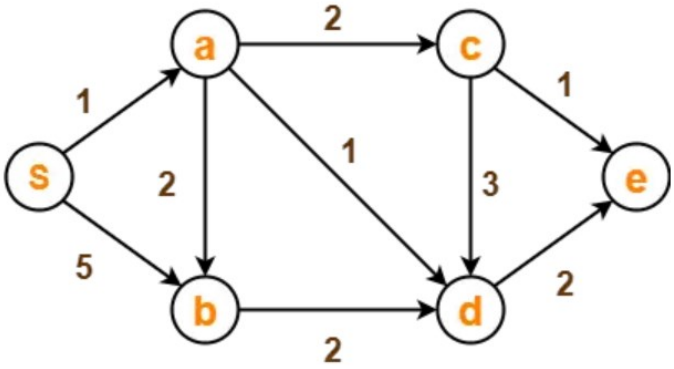
5.	Compute the time complexity of the code snippet given below.
	int beginning = 0;
	for (i_index = 0; i_index < number; i_index++) {
	for (j_index = number; j_index > i_index; j_index--)
	{
	beginning = beginning + i_index + j_index;
	}
	}
		3	**2**	**1, 2, 3, 4, 5, 9, 12**	**1, 2, 12, 16, 17**	**4**

6.	Articulate an algorithm for the creation of an AVL tree. Your algorithm should show the capability of the AVL tree to handle insertions while self-balancing.	**5**	**2**	**1, 2, 3, 4, 5, 9, 12**	**1, 2, 12, 16, 17**	**3**
7.	Consider a max heap with the following elements: 50, 30, 20, 15, 10, 8, 16. Examine the change created in the max heap after inserting the value 60 into the heap.	**2**	**2**	**1, 2, 3, 4, 5, 9, 12**	**1, 2, 12, 16, 17**	**3, 4**

	 Figure 7					
8.	Distinguish between the upper bounding, lower bounding and order functions in terms of time complexity of algorithms using suitable examples to support your answer.	**3**	**3**	**1, 2, 5, 12**	**1, 2, 3**	**4**
9.	Create an algorithm for sorting the elements of the given array in the following manner. The even elements are sorted in increasing order, and the odd elements are sorted in decreasing order. **Input:** arr[] = {0, 1, 2, 3, 4, 5, 6, 7} **Output:** arr[] = {0, 2, 4, 6, 7, 5, 3, 1}	**5**	**3**	**1, 2, 5, 12**	**1, 2, 3**	**6**

10.	Consider the following segment of C-code:
	`int j, n;`
	`j= 1 ;`
	`while (j <= n)`
	`j = j * 2 ;`
	Determine the number of comparisons in executing the loop for any n > 0.
		2	**3**	**1, 2, 5, 12**	**1, 2, 3**	**3**

11.	Illustrate the usage of the Rabin Karp algorithm to search for the pattern 'DATA' in the text, 'I LOVE DATA STRUCTURES'. Showcase the step-wise rolling search for the pattern.	**3**	**4**	**1.2, 3, 12**	**1, 2, 3**	**4**
12.	Devise an algorithm to showcase the balancing of parentheses using a Stack.	**5**	**4**	**1.2, 3, 12**	**1, 2. 3**	**4**
13.	Deduce a search path for the following graph using the Breadth First Search algorithm.	**2**	**5**	**1.2, 3, 12**	**1, 2, 3**	**4**

	 Figure 13					
14.	Compute the shortest path from node S to all other nodes using the Dijkstra's algorithm.  Figure 14	**3**	**5**	**1.2, 3, 12**	**1, 2, 3**	**4**
15.	Compute the shortest path from node A to node J using the A * algorithm for the given graph. Consider the number along the edge as the cost to traverse the edge and the number along the node as the heuristic value to reach that node.	**5**	**5**	**1.2, 3, 12**	**1, 2, 3**	**4**

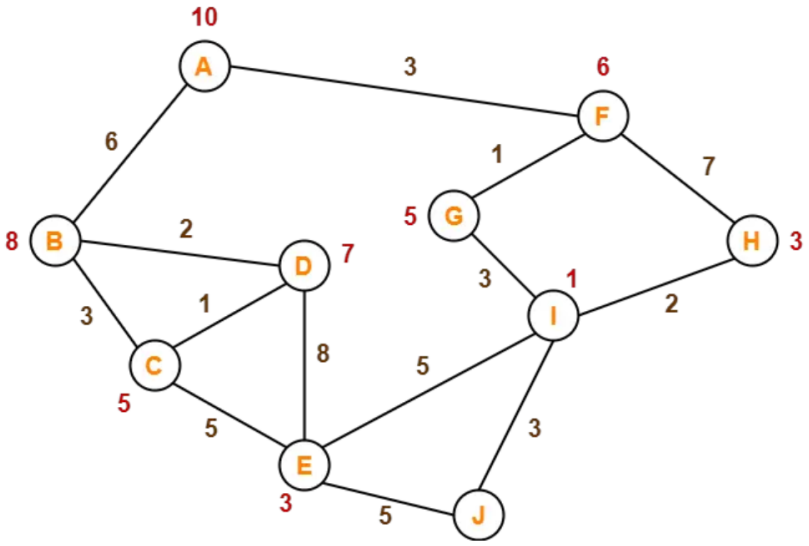


Figure 15