office copy Tan blocks.

Roll Number:_____ Number of Pages: 02

Thapar Institute of Engineering and Technology, Patiala

Department of Computer Science and Engineering

END SEMESTER EXAMINATION

B. E. (Second Year): Sem-I (2022/23) Course Code: UCS301 Course Name: Data Structures

December 12, 2022 Monday, 16:30 Hrs – 19:30 Hrs Time: 3 Hours, M. Marks: 40

Name of Faculty: SP, RMT, RKR, VBP, SMT, Dr. Manisha, Dr. Diwakar

Note: Attempt subparts of a question in sequence at one place. Assume missing data, if any, suitably.

Q1. **(a)** The height of a tree is the length of the longest path from root to leaf. Find the height of the binary tree whose respective postorder and inorder traversals are **8, 9, 6, 7, 4, 5, 2, 3, 1** and **8, 6, 9, 4, 7, 2, 5, 1, 3**. **(4 marks)**

(b) Convert the BST shown in **Fig. 1.** into an AVL tree, showing trees at each intermediate step. **(2 marks)**

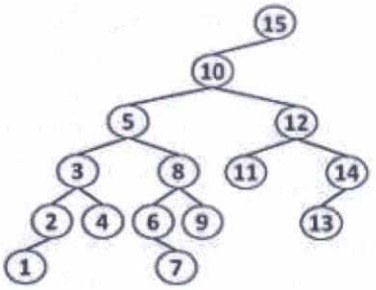

Fig. 1

Q2. **(a)** For the graph shown in **Fig. 2**, determine the sequence in which edges get selected by Prim's minimum spanning tree algorithm. Start from vertex **A**. **(3 marks)**

(b) Execute Dijkstra's single source shortest path algorithm on the graph given in **Fig. 3**. Consider vertex **T** as the starting vertex. **(4 marks)**

	A	B	C	D	E	F	G
A	0	7	0	5	0	0	0
B	7	0	8	9	7	0	0
C	0	8	0	0	5	0	0
D	5	9	0	0	15	6	0
E	0	7	5	15	0	8	9
F	0	0	0	6	8	0	11
G	0	0	0	0	9	11	0

Fig. 2

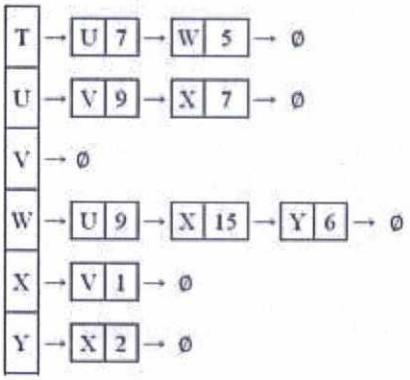

Fig. 3

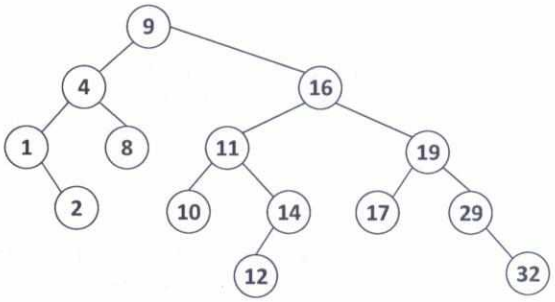

Fig. 4

Q3. **(a)** Build an **AVL tree** by sequentially inserting the keys (**45, 20, 32, 30, 55, 65, 25, 40, 29, 35,** and **50**). After each insertion that includes a rotation, redraw the tree. **(5 marks)**

(b) Delete the node **8** from the AVL tree shown in **Fig. 4** and draw the resulting AVL tree. **(2 marks)**

Q4. Execute BUILD-MAX-HEAP (**Fig. 5**) on the array $A[\,] = \{5, 3, 17, 10, 84, 19, 6, 22, 9\}$. Show all intermediate steps involved in the construction of MAX-HEAP. Also prove that the time complexity of BUILD-MAX-HEAP is $O(n)$, where n is the total number of elements in the heap. **(3 marks) + (3 marks)**

```
BUILD-MAX-HEAP(A)
1  A.heap-size = A.length
2  for i = ⌊A.length/2⌋ downto 1
3      MAX-HEAPIFY(A, i)
```

```
MAX-HEAPIFY(A, i)
1  l = LEFT(i)
2  r = RIGHT(i)
3  if l ≤ A.heap-size and A[l] > A[i]
4      largest = l
5  else largest = i
6  if r ≤ A.heap-size and A[r] > A[largest]
7      largest = r
8  if largest ≠ i
9      exchange A[i] with A[largest]
10     MAX-HEAPIFY(A, largest)
```

Fig. 5

Q5. **(a)** Consider an unsorted array of **13** integers. You have asked to sort it using merge-sort algorithm. Find out the total number of **worst-case** and **best-case** comparisons needed to run the merge sort algorithm? Justify your answers by giving proper example. **(Hint: The total number of comparisons is the exact number of comparisons, not the approximate comparisons.)** **(4 marks)**

(b) You have given an unsorted array having elements **25, 47, 15, 8, 9, 4, 40, 30, 10, 19**, and asked to sort the array using quick sort algorithm. You select the last element **19** as the pivot element. Show the content of the left sub-array and right sub-array after fixing **19** at its correct index assuming that the array starts with the index **zero**. No need to sort the entire array. **(3 marks)**

Q6. **(a)** Consider a hash table shown in **Fig. 6**. Keys are inserted using double hashing collision resolution technique with $h(k,i) = (h_1(k) + ih_2(k)) mod\ m$, $m = 11$, $h_1(k) = k$, and $h_2(k) = 1 + k\ mod\ (m-1)$. Determine the correct indices giving the complete probe sequence for keys **22** and **31**. **(2 marks)**

0	1	2	3	4	5	6	7	8	9	10
88		28		59		17			4	15

Fig. 6

(b) Execute $function\_one\ (node * root)$ in **Fig. 7** on the BST shown in **Fig. 1** and explain the purpose of designing it. **(Hint: Try to draw trees obtained in intermediate steps of the algorithm.)** **(5 marks)**

```
struct node { int key;   node *left, *right; };

int height (node *root); // Returns zero for a single node tree.
node* rightRotate (node *root); // Performs right rotation.
node* leftRotate (node *root); // Performs left rotation.

node* function_two (node *root)
{  // Declare two node pointers p and r. Allocate memory dynamically to pointer r.
   r -> left = NULL;
   r -> right = root;
   p = r;
   while (p -> right) {
      if (p -> right -> left == NULL)     p = p -> right;
      else p -> right = rightRotate(p -> right);
   }
   return r;
}

node* function_three (node *root, int count )
{  // Declare a node pointer p and initialize it to root.
   while (count) {
      p -> right = leftRotate(p -> right);
      p = p -> right;
      --count;
   }
   return root;
}

node* function_one (node *root)
{  root = function_two(root);
   // Declare integers n, t, l, and lc.
   n = height(T);  // T = root
   t = n + 1; l = 0;
   while (t > 1)     {   ++l;   t /= 2;   }
   lc = n + 1 - 2^l;
   if (lc == 0)      lc = 2^(l - 1);
   root = function_three(root,lc);
   n -= lc;
   while (n > 1) {
      n /= 2;
      root = function_three(root,n);
   }
   return T;  // T = root
}
```

Fig. 7

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