EAST WEST UNIVERSITY
Department of Computer Science and Engineering
B.Sc. in Computer Science and Engineering Program
Final Examination, Spring 2021 Semester, Set: A

Course:	**CSE 207- Data Structures, Section-2**
Instructor:	**Dr. Maheen Islam, Associate Professor, CSE Department**
Full Marks:	**50**
Time:	**1 Hour and 30 Minutes**

Note: There are FOUR questions, answer ALL of them.

Problem-1 [10 Marks]

a. Suppose you are a coach of an intra-school football team. You have selected your entire team member. But some more students want to play football. So, you have decided to prepare a waiting list containing name, age, number of matches played, and number of total goals missed of the students. To determine the priority of a student the following formula shown is used.

$$\textit{Priority number} = A + B - C$$

Now **build** a Binary Heap using the priority number where the largest number will have the highest priority.

Name	Age (A)	No. of matches played (B)	No. of goals missed (C)
Rony	15	20	30
Jack	13	15	22
Smith	13	12	10
Paul	12	10	8
Sophia	17	25	10
Jorge	12	10	2
Dave	16	11	7
Eva	14	12	5
pinky	13	13	9
Brown	14	22	14

b. Now consider five students don't want play football anymore and you have decided to select top five students from the waiting list constructed in question no 1(a) according to their priority number. Delete the top five priority students from the Heap and perform necessary operations to **rebuild** the heap after deletion.

Problem-2: [10 Marks]

Construct an AVL tree using the following data entered as a sequential set. Show the balance factors in the resulting tree. **(Show detailed steps).**

70, 57, 33, 77, 72, 82, 28, 48, 51

Next, delete the nodes containing 57 and 72 from the constructed tree. **Draw** each deleted scenario while you delete from the tree and make sure that, the tree created after each delete operation must be balanced.

Problem-3: [10 Marks]

Consider the algorithm given below that construct a binary tree from given Inorder and Preorder traversals.
Algorithm: buildTree()
1) Pick an element from Preorder. Increment a Preorder Index Variable (preIndex in below code) to pick the next element in the next recursive call.
2) Create a new tree node tNode with the data as the picked element.
3) Find the picked element's index in Inorder. Let the index be inIndex.
4) Call buildTree for elements before inIndex and make the built tree as a left subtree of tNode.
5) Call buildTree for elements after inIndex and make the built tree as a right subtree of tNode.
6) return tNode.

Following the algorithm, the C code given below creates a Binary tree from the given traversal sequences.

Find the output of the following programs. **Show** all the calculations (Step-by-step) performed by the function

- `struct node* buildTree(char in[], char pre[], int inStrt, int inEnd)`

```
/* program to construct tree using inorder and preorder traversals */
#include <stdio.h>
#include <stdlib.h>

/* A binary tree node has data, pointer to left child
   and a pointer to right child */
struct node {
    char data;
    struct node* left;
    struct node* right;
};

/* Prototypes for utility functions */
int search(char arr[], int strt, int end, char value);
struct node* newNode(char data);

/* Recursive function to construct binary of size len from Inorder traversal
in[] and Preorder traversal pre[].  Initial values of inStrt and inEnd
should be 0 and len -1.  The function doesn't do any error checking for
cases where inorder and preorder do not form a tree */
```

```
struct node* buildTree(char in[], char pre[], int inStrt, int inEnd)
{
    static int preIndex = 0;

    if (inStrt > inEnd)
        return NULL;

    /* Pick current node from Preorder traversal using preIndex
    and increment preIndex */
    struct node* tNode = newNode(pre[preIndex++]);

    /* If this node has no children then return */
    if (inStrt == inEnd)
        return tNode;

    /* Else find the index of this node in Inorder traversal */
    int inIndex = search(in, inStrt, inEnd, tNode->data);

    /* Using index in Inorder traversal, construct left and
     right subtress */
    tNode->left = buildTree(in, pre, inStrt, inIndex - 1);
    tNode->right = buildTree(in, pre, inIndex + 1, inEnd);

    return tNode;
}

/* UTILITY FUNCTIONS */
/* Function to find index of value in arr[start...end]
   The function assumes that value is present in in[] */
int search(char arr[], int strt, int end, char value)
{
    int i;
    for (i = strt; i <= end; i++) {
        if (arr[i] == value)
            return i;
    }
}

/* Helper function that allocates a new node with the
   given data and NULL left and right pointers. */
struct node* newNode(char data)
{
    struct node* node = (struct node*)malloc(sizeof(struct node));
    node->data = data;
    node->left = NULL;
    node->right = NULL;

    return (node);
}

/* This funtcion is here just to test buildTree() */
void printInorder(struct node* node)
{
    if (node == NULL)
```

```
        return;

    /* first recur on left child */
    printInorder(node->left);

    /* then print the data of node */
    printf("%c ", node->data);

    /* now recur on right child */
    printInorder(node->right);
}

/* Driver program to test above functions */
int main()
{
    char in[] = { 'D', 'B', 'E', 'A', 'F', 'C' };
    char pre[] = { 'A', 'B', 'D', 'E', 'C', 'F' };
    int len = sizeof(in) / sizeof(in[0]);
    struct node* root = buildTree(in, pre, 0, len - 1);

    /* Let us test the built tree by printing Insorder traversal */
    printf("Inorder traversal of the constructed tree is \n");
    printInorder(root);
    getchar();
}
```

Problem-4: [3 +5 + 2 + 5 + 5 = 20 Marks]

(a) **Draw** an undirected connected graph with nine vertices and at least15 edges. The vertices should be called A, B, C, D, E, F, G, H and I. There must be at least 2 paths of length three from D to H and a path of length two from B to I.

(b) **Show** breadth-first traversal of the graph constructed in part (a), where G is the starting node. Visit adjacent nodes in a clockwise ordering from a particular node (12 o'clock position). Show each step to find the traversal sequence using the appropriate data structure.

(c) Convert the graph of part (a) into a weighted graph by assigning weights to each of the edges of the graph; ranging 1 - 20 (randomly pick any value). Do not assign the same weight to more than 2 edges.

(d) **Construct** the minimum spanning tree for the weighted graph constructed in part (c). Also compute the weight. Show the steps you applied to construct the minimum spanning tree.

(e) Use Dijkstra's algorithm to **calculate** minimum cost path (Show detail steps), for the weighted graph constructed in part (c), to travel from node H to all other nodes and show the corresponding routes.