**First Semester 2023-24**

**Data Structures and Algorithms Design (Merged-SEZG519/SSZG519)**

**LAB 3 (Non-Linear Data Structures)**

*Notes: This lab covers the practicals on* *Non-Linear Data Structures. We have mentioned programs in C language in the lab sheet. However, students are free to choose any of the programming languages to develop the solution to lab tasks.*

1. Complete the following code to determine preorder-inorder-postorder of the given binary tree.

|  |
| --- |
| #include<stdio.h>  #include<stdlib.h>  // We are creating struct for the binary tree below  struct node{  int data;  struct node \*left, \*right;  };  // newNode function for initialisation of the newly created node  struct node \*newNode (int item){  struct node \*temporary = (struct node \*) malloc (sizeof (struct node));  temporary->data = item;  temporary->left = temporary->right = NULL;  return temporary;  }  // Here we print the preorder recursively  void preorder (struct node \*root){  ***//Write your code here…***  }  // Here we print the inorder recursively  void inorder (struct node \*root){  ***//Write your code here…***  }  // Here we print the postorder recursively  void postorder (struct node \*root){  ***//Write your code here…***  }  // Basic Program to insert new node at the correct position in BST  struct node \*insert (struct node \*node, int data){  /\* When there no node in the tree(subtree) then create  and return new node using newNode function \*/  if (node == NULL)  return newNode (data);  /\* If not then we recur down the tree to find correct position for insertion \*/  if (data < node->data)  node->left = insert (node->left, data);  else if (data > node->data)  node->right = insert (node->right, data);  return node;  }  void main (){  /\* What our binary search tree looks like really  9  / \  7 14  / \ / \  5 8 11 16 \*/  struct node \*root = NULL;  root = insert (root, 9);  insert (root, 7);  insert (root, 5);  insert (root, 8);  insert (root, 14);  insert (root, 11);  insert (root, 16);  printf ("The preorder is :\n");  preorder (root);  printf ("\nThe inorder is :\n");  inorder (root);  printf ("\nThe postorder is :\n");  postorder(root);  } |

1. Complete the following code to develop heap sort using min heap.

|  |
| --- |
| #include <stdio.h>  /\* function to heapify a subtree. Here 'i' is the index of root node in array a[], and 'n' is the size of heap. \*/  void heapify(int a[], int n, int i)  {  ***//Write your code here…***  }  /\*Function to implement the heap sort\*/  void heapSort(int a[], int n)  {  ***//Write your code here…***  }  /\* function to print the array elements \*/  void printArr(int arr[], int n){  for (int i = 0; i < n; ++i)  {  printf("%d", arr[i]);  printf(" ");  }  }  void main(){  int a[] = {48, 10, 23, 43, 28, 26, 1};  int n = sizeof(a) / sizeof(a[0]);  printf("Before sorting array elements are - \n");  printArr(a, n);  heapSort(a, n);  printf("\nAfter sorting array elements are - \n");  printArr(a, n);  return 0;  } |