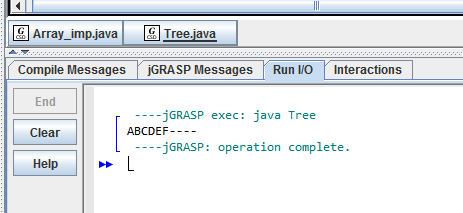
CSCI 321 Computer Science III Summer 2019

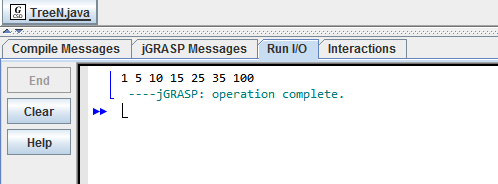
Assignment 2

1. Implement a binary tree using the array-based representation shown in Chapter Slide 33. Code in Java with main class. Use the binary tree in Slide 33 as an input instance. Attach your code and screenshot of the output. See below for example code (details of some methods are missing.)



// JAVA implementation of tree using array   
// numbering starting from 0 to n-1.   
import java.util.\*;   
import java.lang.\*;   
import java.io.\*;   
  
class Tree   
{   
 public static void main(String[] args)   
 {   
 Array\_imp obj = new Array\_imp();   
 obj.Root("A");   
 obj.set\_Left("B", 0);   
 obj.set\_Right("C", 0);   
 obj.set\_Left("D", 1);   
 obj.set\_Right("E", 1);   
 obj.set\_Left("F", 2);   
 obj.print\_Tree();   
  
 /\*Code above is an example of an instance. You need to input the tree shown in slide 33\*/  
 }   
}   
  
  
class Array\_imp {   
 static int root = 0;   
 static String[] str = new String[10];   
   
 /\*create root\*/  
 public void Root(String key)   
 {   
 str[0] = key;   
 }   
  
 /\*create left son of root\*/  
 public void set\_Left(String key, int root)   
 {   
 /\*need your code here\*/  
 int t = (root \* 2)+1;  
   
 if(str[root] == null){   
 System.out.printf("Can't set child at %d, no parent found\n",t);   
 }else{   
 str[t] = key;   
 }   
 }   
  
 /\*create right son of root\*/  
 public void set\_Right(String key, int root)   
 {   
 /\*need your code here\*/  
 int t = (root \* 2)+2;  
   
 if(str[root] == null){   
 System.out.printf("Can't set child at %d, no parent found\n",t);   
 }else{   
 str[t] = key;   
 }   
 }   
  
 public void print\_Tree()   
 {   
 for (int i = 0; i < 10; i++) {   
 if (str[i] != null)   
 System.out.print(str[i]);   
 else  
 System.out.print("-");   
   
 }   
 }   
}

1. Implement a binary tree using the linked structure shown in Chapter Slide 32. Code in Java with main class. Use the binary tree in Slide 33 as an input instance. Attach your code and screenshot of the output. Hint: See the sample code above as a reference.



import java.util.\*;  
  
class TreeN   
{  
 // A linked list node  
 int data;  
   
 TreeN left;  
   
 TreeN right;  
   
 TreeN(int data)   
 {  
   
 left = null;  
 right = null;  
   
 this.data = data;  
 }  
}  
  
class Tree // Main class  
{  
  
 TreeN fun(TreeN root, int data)   
 { // create binary tree  
   
 if (root == null)   
 {  
 return new TreeN(data);  
 }  
 if (data > root.data)   
 {  
 root.right = fun(root.right, data);  
 }  
 else if (data < root.data)   
 {  
 root.left = fun(root.left, data);  
 }  
 else   
 {   
 return root;  
 }  
   
 return root;  
 }  
  
  
 void display(TreeN head) //display binary tree  
 {  
 if (head != null)  
 {  
 display(head.left);  
 System.out.print(head.data + " ");  
 display(head.right);  
 }  
 }  
 public static void main(String[] args)  
 {  
 TreeN head=null;  
 Tree t = new Tree();  
   
 head = t.fun(head , 10);  
 head = t.fun(head , 5);  
 head = t.fun(head , 15);  
 head = t.fun(head , 25);  
 head = t.fun(head , 1);  
 head = t.fun(head , 35);  
 head = t.fun(head , 100);  
   
 t.display(head); // Inorder Traversal  
   
 }  
}