|  |
| --- |
| **Computer Engineering Department - ITU** |
| **CE200L: Data Structures & Algorithms Lab** |

|  |  |
| --- | --- |
| **Course Instructor: Usama Bin Shakeel** | **Dated: 25/10/2022** |
| **Teaching Assistant: Muhammad Sufyan Ashraf** | **Semester: Fall 2022** |
| **Lab Engineer: Nadir Abbas** | **Batch: BSCE2021** |

# **Lab 9A. Complete Tree Implementation with Pointers**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Roll number** | **Report**  **(out of 100)** | **Scaled to 10** | **Total**  **(out of 10)** |
|  |  |  |  |  |

Checked on: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## **Objective**

The objective of this lab is to provide the knowledge of basic data structures and their implementations.

## **Equipment and Component**

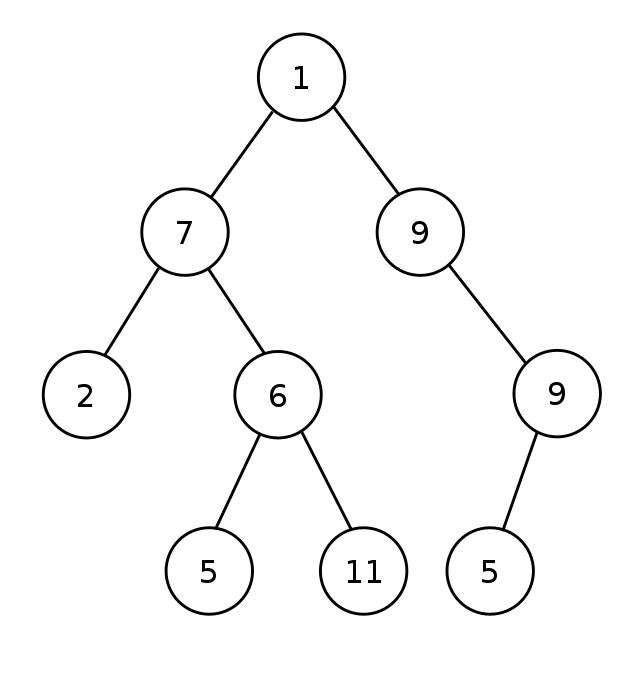
|  |  |  |
| --- | --- | --- |
| **Component Description** | **Value** | **Quantity** |
| Computer | Available in lab | 1 |

## **Conduct of Lab**

1. Students are required to perform this experiment individually.
2. In case the lab experiment is not understood, the students are advised to seek help from the course instructor, lab engineers, assigned teaching assistants (TA) and lab attendants.

## **Theory and Background**

In computer science, A **full binary tree** (sometimes proper binary tree or 2-tree) is a tree in which every node other than the leaves has two children.



In computer science, **Arrays** are used to store multiple values in a single variable, instead of declaring separate variables for each value. To declare an array, define the variable type, specify the name of the array followed by square brackets and specify the number of elements it should store: string cars [4];

**Templates** are a feature of the C++ programming language that allows functions and classes to operate with generic types. This allows a function or class to work on many different data types without being rewritten for each one.

**Lab Task**

**Task A**

As you have implemented the binary tree before, Now implement a complete binary tree with pointers. Two pointers for each tree node for two children Check using them if the tree is complete or not. Implement the following functions:

* Insert
* Update
* Delete
* Display
* Check

Make all necessary functions and handle all corner cases. Make a menu-driven program.

|  |
| --- |
| // Paste your code here  Function.h:  // // Created by Lenovo on 10/25/2022. //  #ifndef INC\_2022\_FALL\_CE\_DSA\_WEEK9\_LABTASK\_A\_BSCE21012\_FUNCTIONS\_H #define INC\_2022\_FALL\_CE\_DSA\_WEEK9\_LABTASK\_A\_BSCE21012\_FUNCTIONS\_H  #endif //INC\_2022\_FALL\_CE\_DSA\_WEEK9\_LABTASK\_A\_BSCE21012\_FUNCTIONS\_H  #include <iostream>  using namespace std;  class node { public:  int data;  node \*right;  node \*left;  node \*parent;   node() {  data = 0;  left = nullptr;  right = nullptr;  parent = nullptr;  }   node(int val) {  data = val;  right = nullptr;  left = nullptr;  parent = nullptr;  } };  class linklist { public:  node \*root;   linklist() {  root = nullptr;  }  node \*Insert(node \*temp, int key) {  if (temp == nullptr) {  temp = new node;  temp->data = key;  temp->left = nullptr;  temp->right = nullptr;  temp->parent = nullptr;  }   else if (temp->data < key) {  temp->right = Insert(temp->right, key);  temp->right->parent = temp;  }   else {  temp->left = Insert(temp->left, key);  temp->left->parent = temp;  }  return temp;  }  void print( node \*temp)  {  if(temp == nullptr)  return;  print(temp->left);  cout<<temp->data;  print(temp->right);  }  node\* deleteNode(node\* temp, int val) {  if (temp == nullptr) {  return nullptr;  }  temp->left = deleteNode(temp->left, val);  temp->right = deleteNode(temp->right, val);  // checking the current node data with x  if (temp->data == val && temp->left == nullptr && temp->right == nullptr) {  // deleting the node  return nullptr;  }  return root;  }   };  // // Created by Lenovo on 10/25/2022. // #include <iostream> #include "Functions.h"  using namespace std;  int main() {  node n;  node n1(0);  linklist l;  int opt;  int val;  do{  cout<<"1.INSERT."<<endl;  cout<<"2.DELETE."<<endl;  cout<<"3.EXIT."<<endl;  cin>>opt;  node \*temp = nullptr;  if(opt==1){  cout << "ENTER VALUE = ";  cin >> val;  temp = l.Insert(temp, val);  l.print(temp);  cout<<endl;  }  if(opt==2){  l.deleteNode(temp, val);  cout<<"DELETED NODE = ";  l.print(temp);  }  if(opt==3){  cout<<"YOU CHOOSE TO EXIT."<<endl;  exit(3);  }  }while(opt>=1 && opt<=3);   }  // Paste your output here |

#### **Assessment Rubric for Lab**

**Method for assessment:**

Lab reports and instructor observation during lab sessions. Outcome assessed:

a. Ability to conduct experiments, as well as to analyze and interpret data (P) b. Ability to function on multi-disciplinary teams (A)

c. Ability to use the techniques, skills, and modern engineering tools necessary for engineering practice (P)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Performance metric** | **Task** | **CLO** | **Description** | **Max marks** | **Exceeds expectation** | **Meets expectation** | **Does not meet expectation** | **Obtained marks** |
| 1. Realization of experiment (a) | 1 | 1 | Functionality | 40 | Executes without errors excellent user prompts, good use of symbols, spacing in output. Through testing has been completed (35-40) | Executes without errors, user prompts are understandable, minimum use of symbols or spacing in output. Some testing has been completed (20-34) | Does not execute due to syntax errors, runtime errors, user prompts are misleading or non-existent. No testing has been completed (0-19) |  |
| 2. Teamwork (b) | 1 | 3 | Group Performance | 5 | Actively engages and cooperates with other group member(s) in effective manner (4-5) | Cooperates with other group member(s) in a reasonable manner but conduct can be improved (2-3) | Distracts or discourages other group members from conducting the experiment (0-1) |  |
| 3. Conducting experiment (a, c) | 1 | 1 | On Spot Changes | 10 | Able to make changes (8-10) | Partially able to make changes (5-7) | Unable to make changes (0-4) |  |
| 1 | 1 | Viva | 10 | Answered all questions (8-10) | Few incorrect answers (5-7) | Unable to answer all questions (0-4) |  |
| 4. Laboratory safety and disciplinary rules (a) | 1 | 3 | Code commenting | 5 | Comments are added and does help the reader to understand the code (4-5) | Comments are added and does not help the reader to understand the code (2-3) | Comments are not added (0-1) |  |
| 5. Data collection (c) | 1 | 3 | Code Structure | 5 | Excellent use of white space, creatively organized work, excellent use of variables and constants, correct identifiers for constants, No line-wrap (4-5) | Includes name, and assignment, white space makes the program fairly easy to read. Title, organized work, good use of variables (2-3) | Poor use of white space (indentation, blank lines) making code hard to read, disorganized and messy (0-1) |  |
| 6. Data analysis (a, c) | 1 | 4 | Algorithm | 20 | Solution is efficient, easy to understand, and maintain (15-20) | A logical solution that is easy to follow but it is not the most efficient (6-14) | A difficult and inefficient solution (0-5) |  |
| 7. Computer use (c) | 1 | 2 | Documentation & Github Submissions | 5 | Timely (4-5) | Late (2-3) | Not done (0-1) |  |
|  | Max Marks (total): | | | 100 | Obtained Marks (total): | | |  |

Lab Engineer Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_