|  |  |  |  |
| --- | --- | --- | --- |
| PICTLOGO | PUNE INSTITUTE OF COMPUTER TECHNOLOGY  PUNE - 411043 | | |
| Department of Electronics & Telecommunication | | |
| ASSESMENT YEAR: 2024-2025 | CLASS: SE | |
| SUBJECT: DATA STRUCTURES | | |
| **EXPT No:** | LAB Ref: SE/2024-25/ | | Starting date: |
|  | Roll No: 22168 | | Submission date: |
| **Title:** | **Expt.-8: Creation of BST** | | |
| **Problem Statement** | Implement Binary search tree and perform operations. such as Create,  search, and recursive traversal (In order, preorder and post order). | | |
| Refer lab manual for below | | | |
| **Prerequisites:** | Software - DEVC++ IDE/GCC | | |
|  | | |
|  | | |
|  | | |
| **Objectives:** | 1) Open DEVC++ IDE/GCC, create a new file in the folder named with | | |
| roll number, Write the program (Compile & Run). | | |
| 2) Verify output generated for various test cases as suggested by instructor. | | |
| 3. Write in journals and verify from instructor weekly. | | |
| **Theory:** | | | |
|  | Write algorithm and besides it example for each, attach both side ruled pages in case required  On Ruled Pages | | |
| **Flow-chart**  **ERRORS** | **(Extra sheet may be attached)**  **Here are some errors:**  ** Memory allocation failure: malloc() returns NULL.**  ** Uninitialized root pointer: Accessing an uninitialized root.**  ** Segmentation fault during traversal: Traversing an empty tree.**  ** Invalid user input for traversal option: User enters an invalid choice.**  ** Logical error in insertion: Using <= or >= leads to duplicates in the BST.** | | |
| **REMEDY** | ** Memory allocation failure: Check malloc() for NULL and handle it with an error message or exit.**  ** Uninitialized root pointer: Initialize root = NULL before operations.**  ** Segmentation fault during traversal: Check if root == NULL before traversing.**  ** Invalid user input for traversal: Add a default case to handle invalid inputs.**  ** Logical error in insertion: Use strict comparisons (<, >) to avoid duplicates.Bottom of Form** | | |
|  | | | |
| **CONCLUSION:** | | | |
|  | **The Binary Search Tree (BST) implementation in C effectively organizes and** | | |
| **manages data, supporting operations like creation, insertion, searching, and** | | |
| **various traversal methods (In-order, Pre-order, Post-order). Addressing common** | | |
| **errors ensures robustness, while user-friendly prompts enhance interactivity** | | |
| **Overall, this program serves as a valuable tool for understanding tree data** | | |
| **structures and their functionalities.** | | |
|  | | |
| **REFERENCES: refer lab manual for the same** | | | |
|  | **1. Ellis Horowitz, Sartaj Sahani, “Fundamentals of Data Structures”,** | | |
| **Galgotia books.** | | |
| **2. Richard F. Gilberg and Behrouz A. Forouzan, Data Structures A Pseudo** | | |
| **code approach with C, cengage learning, 2nd edition.** | | |
| **3. Yashvant Kanetkar-Understanding Pointers in C BPB publications 3rd** | | |
| **Edition.** | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Continuous Assessment for DS AY: 2024-25** | | | |
| **RPP (5)** | **SPO (5)** | **Total (10)** | **Signature:** |
|  |  |  | **Assessed By: Dr. V. B. Vaijapurkar** |
| **Start date** | **Submission date** | | **Date:** |
|  |  | |  |
| **\*Regularity, Punctuality, performance**  **\*Submission, Presentation, orals** | | | |