Report on

"Generic Binary Search Tree on Hard Disk "
project-id = 13

Submitted in partial fulfillment of the requirements for ***Sem VI***

Generic Programming

Bachelor of Technology
in
Computer Science & Engineering

Submitted by:

Rohit Vishwakarma PES1201800152

Under the guidance of

Prof. Nagbhushan Satya Kumar
<Professor>
PES University, Bengaluru

January – May 2021

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING
PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India

TABLE OF CONTENTS

Introduction

Generic Binary Search Tree on Hardisk

This project is aimed at creating a generic Binary Search Tree library which creates generic binary search tree on the hard disk which is dumped into a file and the entire tree is generic and the pointers of the tree nodes are managed by carefully using file offsets instead of pointers this is the way we deal with the task of storing the Binary Search Tree .

This library provides us with the following

1. Create tree nodes of generic type.
2. Insert and Delete nodes.
3. Dump them onto a file.
4. Name that file
5. Extract inorder traversal of the tree .
6. Extract preorder traversal of the tree.
7. Extract postorder traversal of the tree.
8. Get the traversals in the form of the vector.

The library is inside the Namespace of tree
the name of the header file is tree.h

FAQs

Q1) Why to use this library tree.h ?
Ans : Because if you want to access your generic - binary search tree even after you terminate your program.

Q2) What kind of Data types can this generic work with ?
Ans : Any data type which supports
(i) < operator
(ii) > operator
(iii) == operator
(iv) != operator
(v) copy constructor.
(vi) copy assignment operator.
(vii) put to operator or operator<<

Q3) How to use it
Ans : Include the header file and start using it !

IMPLEMENTATION DETAILS

I have built a generic class for tree and its nodes which can be used to create binary search tree of generic type .

The Each tree node is having field like .

T key = > An object of type T.

leftOffset => The offset of left child if(null then -1)

rightOffset => The offset of right child if(null then -1)

There are three methods available in a tree

1. *void insert(T key);* *(Average Theta (log n) , Worst case O(n))*

   This method inserts node of type T into the tree.

2. *void delete_key(T key);* *(Average Theat (log n), Worst case O(n))*

   The method deletes the given node from the tree and joins the inorder successor if the node is the parent of two children .

3. *void display_inorder()*

   => This method prints the inorder traversal of the tree and its also the sorted order .

4. *void display_preorder()*

   This method prints the preorder traversal of the tree.

5. *void display_postorder()*

   This method prints the postorder traversal of the tree.

6. *vector<T> get_inorder()*

   This method returns a vector of type T with inorder traversal of the tree.

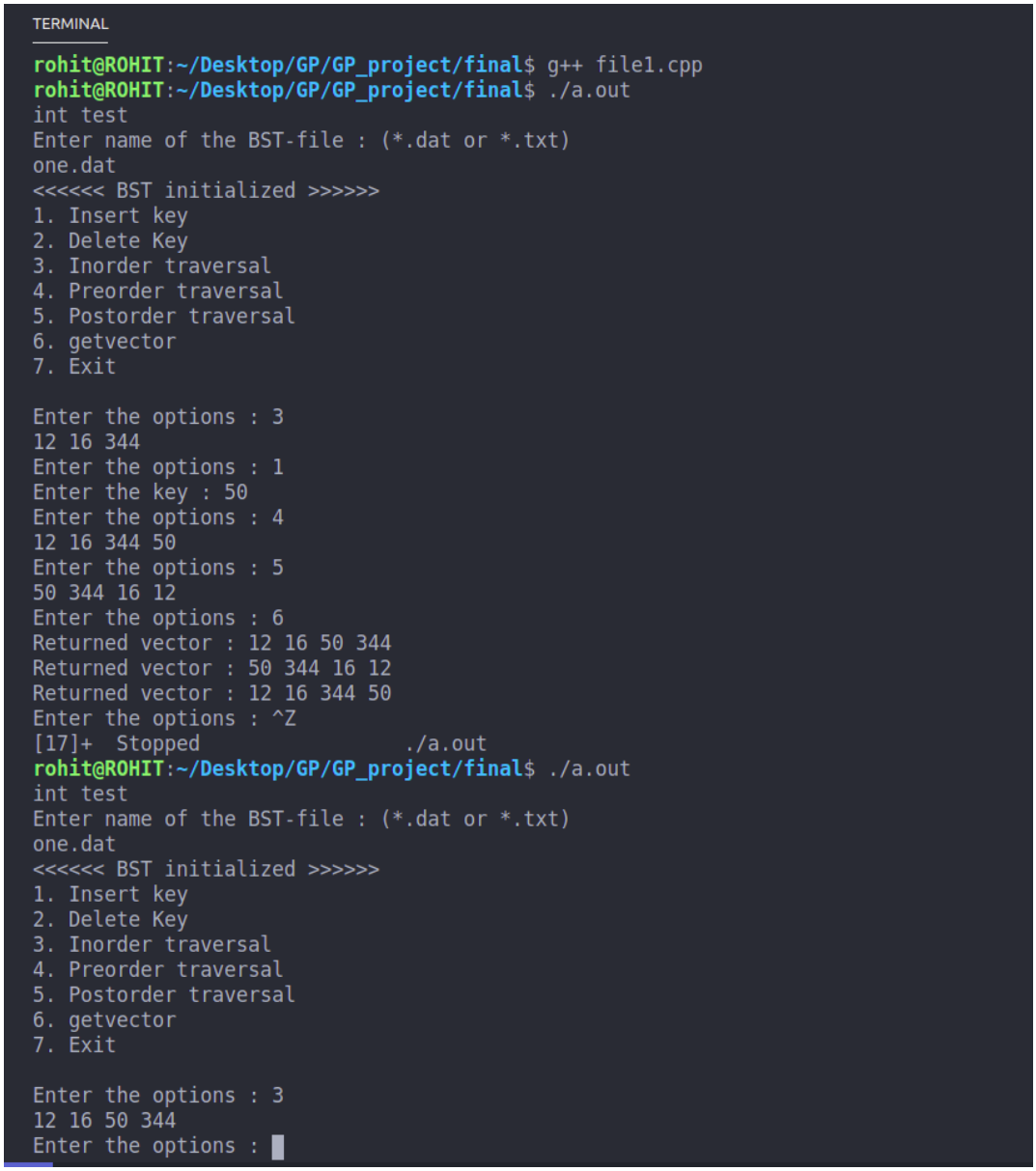
7. *vector<T> get_preorder()*

   This method returns a vector of type T with preorder traversal of the tree.

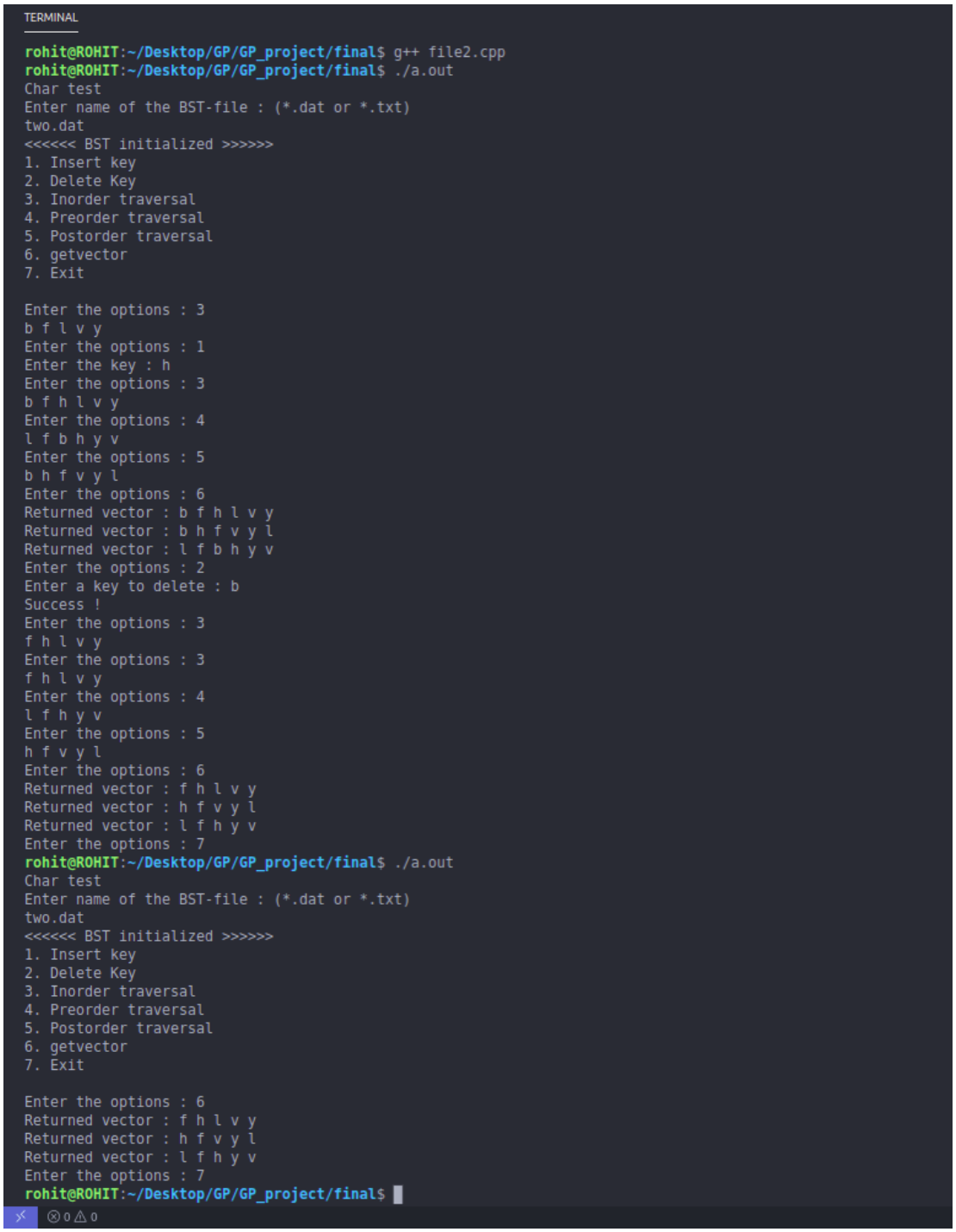
8. *vector<T> get_postorder()*

   This method returns a vector of type T with postorder traversal of the tree.

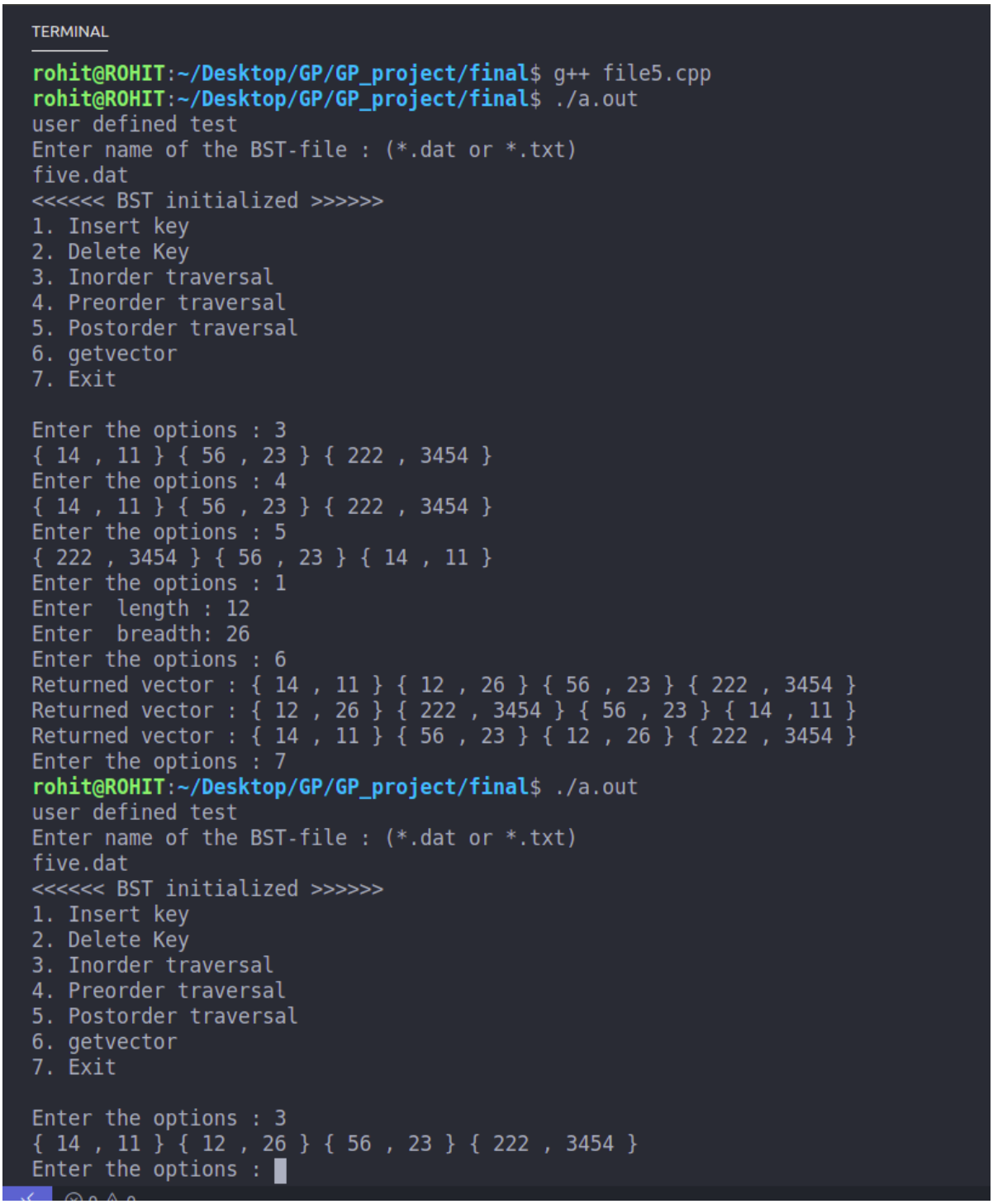
SNAPSHOTS 1 (tested with int data)



SNAPSHOT 2 (tested with char data)



SNAPSHOT 3 (on a user defined datatype)



RESULTS AND FURTHUR ENHANCEMENT

Results:

I have used all knowledge from the Generic Programming Course and have implemented a library in C++ for Generic Binary Search Tree which generates Binary Search trees dumped into files and which are generic in nature which could be used again in the program after terminating it in some other program .

Future Enhancement :

This library could be made more robust and powerful by implementing various algorithms and for more complex data types .