Experiment - 10

Student Name: Pankaj Singh Kanyal **UID:** 20BCS6668
Branch: AIML **Section/Group:** AIML 4 B
Semester: 5th **Date of Performance:** / /2022
Subject Name: Advanced Programming Lab **Subject Code:** 20CSP-334

1. AIM:

Demonstrate insert, delete and search in Treap

2. Apparatus:

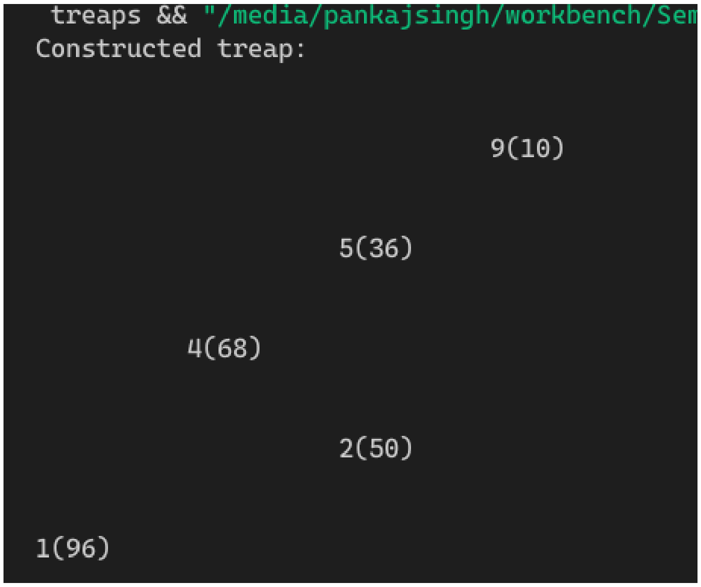
- Texeditor
- Laptop / PC with C++ compiler

3. Program/Code

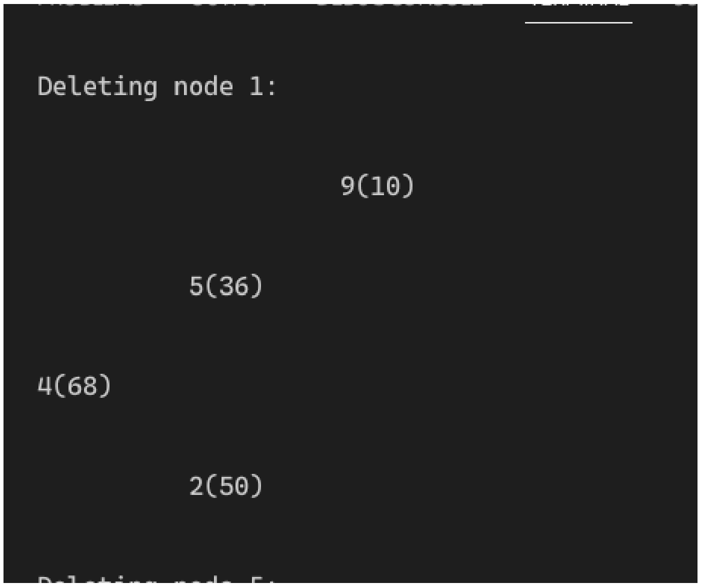
```
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;
struct TreapNode
{
   int data;
   int priority;
   TreapNode* left, *right;
   TreapNode(int data)
   {
       this->data = data;
       this->priority = rand() % 100;
       this->left = this->right = nullptr;
   }
};
```

```
void rotateLeft(TreapNode* &root)
{
    TreapNode* R = root->right;
    TreapNode* X = root->right->left;
    R->left = root;
    root->right = X;
    root = R;
}

void rotateRight(TreapNode* &root)
{
    TreapNode* L = root->left;
    TreapNode* Y = root->left->right;

    L->right = root;
    root->left = Y;

    root = L;
}

void insertNode(TreapNode* &root, int data)
{
    if (root == nullptr)
    {
        root = new TreapNode(data);
        return;
    }

    if (data < root->data)
    {
        insertNode(root->left, data);
        if (root->left != nullptr && root->left->priority > root->priority) {
            rotateRight(root);
        }
    }
    else {
        insertNode(root->right, data);
        if (root->right != nullptr && root->right->priority > root->priority) {
            rotateLeft(root);
        }
    }
}

bool searchNode(TreapNode* root, int key)
{
    if (root == nullptr) {
        return false;
    }
    if (root->data == key) {
        return true;
    }

    if (key < root->data) {
        return searchNode(root->left, key);
    }

    return searchNode(root->right, key);
}

void deleteNode(TreapNode* &root, int key)
{
    if (root == nullptr) {
        return;
    }

    if (key < root->data) {
        deleteNode(root->left, key);
    }

    else if (key > root->data) {
        deleteNode(root->right, key);
    }
    // if the key is found
    else {
        // Case 1: node to be deleted has no children (it is a leaf node)
        if (root->left == nullptr && root->right == nullptr)
        {
            delete root;
            root = nullptr;
        }
        // Case 2: node to be deleted has two children
        else if (root->left && root->right)
        {
            if (root->left->priority < root->right->priority)
            {
                rotateLeft(root);

                deleteNode(root->left, key);
            }
            else {
                rotateRight(root);
                deleteNode(root->right, key);
            }
        }
        // Case 3: node to be deleted has only one child
        else {
            TreapNode* child = (root->left)? root->left: root->right;
            TreapNode* curr = root;
            root = child;
            delete curr;
        }
    }
}

void printTreap(TreapNode *root, int space = 0, int height = 10)
{
    if (root == nullptr) {
        return;
    }
    // increase distance between levels
    space += height;
    // print the right child first
    printTreap(root->right, space);
    cout << endl;
    for (int i = height; i < space; i++) {
        cout << ' ';
    }
    cout << root->data << "(" << root->priority << ")\n";
    cout << endl;
    printTreap(root->left, space);
}

int main()
{
    // Treap keys
    int keys[] = { 5, 2, 1, 4, 9, 8, 10 };
    int n = sizeof(keys)/sizeof(int);
    // Construct a treap
    TreapNode* root = nullptr;
    srand(time(nullptr));
    for (int key: keys) {
        insertNode(root, key);
    }
    cout << "Constructed treap:\n\n";
    printTreap(root);
    cout << "\nDeleting node 1:\n\n";
    deleteNode(root, 1);
    printTreap(root);
    cout << "\nDeleting node 5:\n\n";
    deleteNode(root, 5);
    printTreap(root);
    cout << "\nDeleting node 9:\n\n";
    deleteNode(root, 9);
    printTreap(root);
    return 0;
}
```

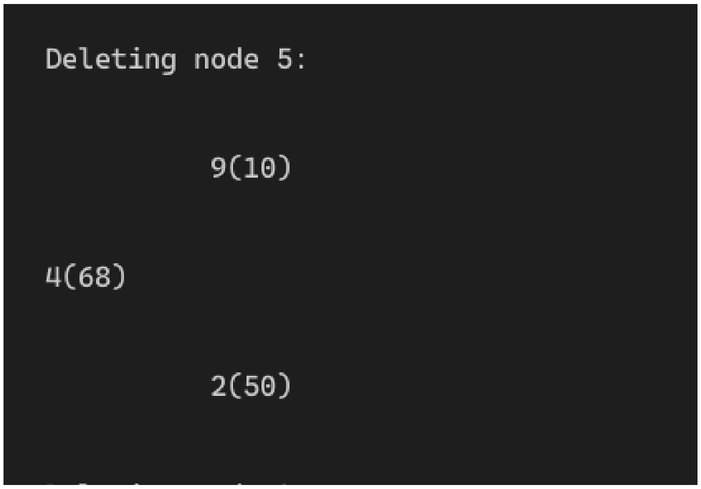
6. Output



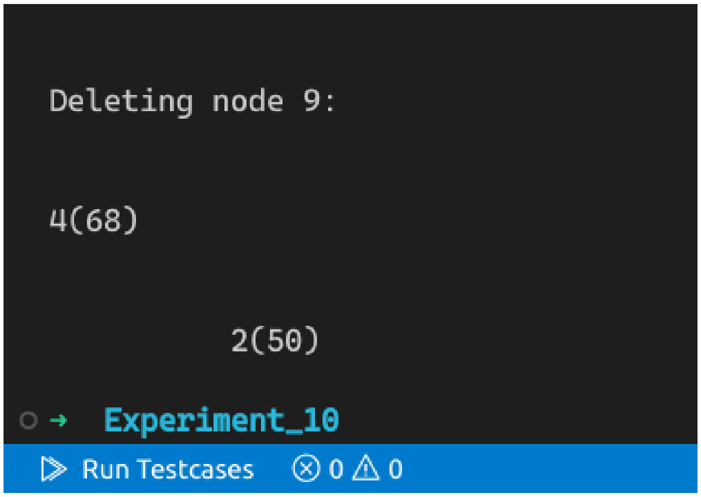
```
 treaps && "/media/pankajsingh/workbench/Sem
Constructed treap:

                              9(10)

                    5(36)

          4(68)

                    2(50)

1(96)
```



```
Deleting node 1:

                    9(10)

          5(36)

4(68)

          2(50)
```



```
Deleting node 5:

          9(10)

4(68)

          2(50)
```



```
Deleting node 9:

4(68)

          2(50)
```

7. Learning Outcomes:

1. Learn to implement construct, insert,delete in a treap

2. Learn about BST and Heap.

3. Learned to write a program for the above problem.

4. Learned to use Clion IDE.

Evaluation Grid (To be created as per the SOP and Assessment guidelines by the faculty):

Sr. No.	Parameters	Marks Obtained	Maximum Marks
1.			
2.			
3.			