Page No.
Date

Sumanth.KV
1BM18CS112
5th C

```
struct Node {
    int data;
    Node *parent;
    Node *left;
    Node *right;
    int color;
};
typedef Node * Nodeptr;

class RedBlackTree {
  private:
    Nodeptr root;
    Nodeptr TNULL;

    void initializeNULLNode(Nodeptr node,
                            Nodeptr parent)
    {
      node->data = 0;
      node->parent = parent;
      node->left = nullptr;
      node->right = nullptr;
      node->color = 0;
    }

    void insertFix(Nodeptr k){
      Nodeptr u;
      while(k->parent->color == 1){
        if(k->parent == k->parent->parent->right){
          u = k->parent->parent->left;
```

```
if (u->color == 1) {
    u->color = 0;
    k->parent->color = 0;
    k->parent->parent->color = 1;
    k = k->parent->parent;
}
else {
    if (k == k->parent->left) {
        k = k->parent;
        rightRotate(k);
    }
    k->parent->color = 0;
    k->parent->parent->color = 1;
    leftRotate(k->parent->parent);
}
}
else {
    u = k->parent->parent->right;
    if (u->color == 1) {
        u->color = 0;
        k->parent->color = 0;
        k->parent->parent->color = 1;
        k = k->parent->parent;
    }
    else {
        if (k == k->parent->right) {
            k = k->parent;
            leftRotate(k);
        }
        k->parent->color = 0;
        k->parent->parent->color = 1;
        rightRotate(k->parent->parent);
    }
}
```

```
        if (k == root){
            break;
        }
    }
    root->color = 0;
}

void printHelper(NodePtr root, string indent,
                 bool last) {
    if (root != TNULL) {
        cout << indent;
        if (last) {
            cout << "R----";
            indent += "     ";
        }
        else {
            cout << "L----";
            indent += "|    ";
        }

        string sColor = root->color ? "RED" : "Black";
        cout << root->data << "(" << sColor << ")";
        printHelper(root->left, indent, false);
        printHelper(root->right, indent, true);
    }
}

public:
RedBlackTree() {
    TNULL = new Node;
    TNULL->color = 0;
    TNULL->left = nullptr;
    TNULL->right = nullptr;
    root = TNULL;
}
```

```
void insert(int key) {
    NodePtr node = new Node;
    node->parent = nullptr;
    node->data = key;
    node->left = TNULL;
    node->right = TNULL;
    node->color = 1;

    NodePtr y = nullptr;
    NodePtr x = this->root;

    while (x != TNULL) {
        y = x;
        if (node->data < x->data) {
            x = x->left;
        }
        else {
            x = x->right;
        }
    }
    node->parent = y;
    if (y == nullptr) {
        root = node;
    }
    else if (node->data < y->data) {
        y->left = node;
    }
    else {
        y->right = node;
    }
    if (node->parent == nullptr) {
        node->color = 0;
        return;
    }
```

```
if (node->parent->parent == nullptr) {
    return;
}
insertFix(node);
}
```

(5) Sumanth-ler