CLASSMATE

Date

ARPANA MR
1BM18CS047

18/11/2020

PROGRAM 07

Q) WAP to implement insertion in Red Black Tree

```
struct RBNode{
    int data;
    RBNode* parent;
    RBNode* left;
    RBNode* right;
    int color;
};

class RBTree{
private:
RBNode* root;
RBNode* Tnull;

void initNode(RBNode* node, RBNode* parent){
    node->parent=parent;
    node->left=NULL;
    node->right=NULL;
    node->data=0;
    node->color=0;
```

classmate

ARPANA MR

Page

18/11/2020

```
void rbTransplant(RBNode* x, RBNode* y)
{
  if (x->parent == NULL){
      root = y;
  }
  else if (x == x->parent->left){
     x->parent->left = y; }
  else{
     x->parent->right = y;
  }
  y->parent = x->parent;
}


void Balance(RBNode* n){
  RBNode* m;

  while(n->parent->color == 1){
    if (n->parent == n->parent->parent->right)
{
     m = n->parent->parent->left;
     if (m->color == 1) {
       m->color = 0
       n->parent->color = 0;
       n->parent->parent->color = 1;
```

IBM18CS141

16/11/2020

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

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

void printN(RBNode* root, string s, bool last)
  {
    if (root!=Tnull){
    cout<<s;
    if(last){
      cout<<"Right----";
      s+="    ";
    }else{
      cout<<"Left----";
      s+="|   ";
    }
    string NodeColor = root->color?"Red":"Black";
    cout<<root->data<<"("<<NodeColor<<")"<<endl;
    printN(root->left, s, false);
    printN(root->right, s, true);
  }}

public:
  RedBlackTree(){
    Tnull=new Node;
```

1BM18CS141
Aishwarya

```
Tnull -> left = NULL;
Tnull -> right = NULL;
Tnull -> color = 0;
root = Tnull;
}

RBNode* minimum(RBNode* node){
    while(node->left != Tnull) node = node->left;
    return node;}
RBNode* maximum

void leftRotate(RBNode* n){
    RBNode* y = n->right;
    n->right = y->left;
    if(y->left != Tnull) y->left->parent = n;
    y->parent = n->parent;
    if(n->parent == NULL) this->root = y;
    else if(n == n->parent->left) n->parent->left = y;
    else n->parent->right = y;
    y->left = n;
    n->parent = y;}
```

1BM18CS144
Arpeena

```
void rightRotate(RBNode* n){
    RBNode* y = n->left;
    n->left = y->right;
    if(y->right != Tnull){
        y->right->parent = n;}
    y->parent = n->parent;
    if(n->parent == NULL) this->root = y;
    else if(n == n->parent->right) n->parent->right = y;
    else n->parent->left = y;
    y->right = n;
    n->parent = y;}

void insert(int value){
    RBNode* = new RBNode; node->parent = NULL;
    node->left = Tnull; node->right = Tnull;
    node->data = value; node->color = 1;
    RBNode* y = NULL; RBNode* n = this->root;
    while(n != Tnull){
        y = x; if(node->data < n->data){
            n = n->left;}
        else{ n = n->right;}}
    node->parent = y;
    if(y == NULL){ root = node;
    else if(node->data < y->data) y->left = node
    else y->right = node;
}
```

1RM18CS147
Arkana

~~Vdstest~~ Fi Balance (node);
RBNode* getRoot(){
return this→root; }
}
~~void pro~~