Program-8: Design, Develop a program in C to implement various operations on Red-Black Tree.

```
#include<stdio.h>
#include<stdlib.h>
typedef struct NODE{
   int key;
   char color;
   struct NODE *left, *right,*parent;
}NODE;
NODE *root = NULL;

void leftRotate(NODE *x){
   NODE *y;
   y = x->right;
   x->right = y->left;
   if( y->left != NULL)
   {
      y->left->parent = x;
   }
   y->parent = x->parent;
   if( x->parent == NULL){
      root = y;
   }
   else if((x->parent->left!=NULL) && (x->key == x->parent->left->key))
   {
      x->parent->left = y;
   }
   else x->parent->right = y;
   y->left = x; x->parent = y; return;
}

void rightRotate(NODE *y){
   NODE *x;
   x = y->left;
   y->left = x->right;
   if ( x->right != NULL)
   {
      x->right->parent = y;
   }
   x->parent = y->parent;
   if( y->parent == NULL)
   {
      root = x;
   }
   else if((y->parent->left!=NULL)&& (y->key == y->parent->left->key))
   {
      y->parent->left = x;
   }
   else
   y->parent->right = x;
   x->right = y; y->parent = x;
   return;
}
void colorinsert(NODE *z){
   NODE *y=NULL;
   while ((z->parent != NULL) && (z->parent->color == 'r'))
   {
      if ((z->parent->parent->left != NULL) && (z->parent->key == z->parent->parent-
>left->key))
      {
         if(z->parent->parent->right!=NULL)
            y = z->parent->parent->right;
         if ((y!=NULL) && (y->color == 'r'))
         {
            z->parent->color = 'b';
            y->color = 'b';
            z->parent->parent->color = 'r';
            if(z->parent->parent!=NULL)
               z = z->parent->parent;
         }
         else
         {
            if ((z->parent->right != NULL) && (z->key == z->parent->right->key))
            {
               z = z->parent;
               leftRotate(z);
            }
            z->parent->color = 'b';
            z->parent->parent->color = 'r';
            rightRotate(z->parent->parent);
         }
      }
      else
      {
         if(z->parent->parent->left!=NULL)
            y = z->parent->parent->left;
         if ((y!=NULL) && (y->color == 'r'))
         {
            z->parent->color = 'b';
            y->color = 'b';
            z->parent->parent->color = 'r';
            if(z->parent->parent!=NULL)
               z = z->parent->parent;
         }
         else
         {
            if ((z->parent->left != NULL) && (z->key == z->parent->left->key))
            {
               z = z->parent;
               rightRotate(z);
            }
            z->parent->color = 'b';
            z->parent->parent->color = 'r';
            leftRotate(z->parent->parent);
         }
      }
   }
   root->color = 'b';
}

void inorder(NODE* root){
   NODE* temp = root;
   if (temp != NULL)
   {
      inorder(temp->left);
      printf(" %d-%c ",temp->key,temp->color);
      inorder(temp->right);
   }
   return;
}

void insert(int val){
   NODE *cur, *prev;
   NODE *z = (NODE*)malloc(sizeof(NODE));
   z->key = val;
   z->left = NULL;
   z->right = NULL;
   z->color = 'r';
   cur=root;
   if ( root == NULL )
   {
      root = z;
      root->color = 'b';
      return;
   }
   while ( cur != NULL)
   {
        prev = cur;
        if ( z->key < cur->key)
        {
           cur = cur->left;
        }
        else
           cur = cur->right;
    }
    z->parent = prev;
    if ( prev == NULL)
    {
        root = z;
    }
    else if( z->key < prev->key )
    {
        prev->left = z;
    }
    else{
        prev->right = z;
    }
    colorinsert(z);
    return;
}
```

int main()

```
{
    int choice, val;
    while(1)
    {
    printf("\nRed Black Tree Menu - \nEnter your choice :\n1:Insert\n2:Traversal
\n3:Exit\n");
        scanf("%d",&choice);
        switch(choice)
        {
           case 1:printf("Enter the integer you want to add : ");
              scanf("%d",&val);
              insert(val);
           break;
           case 2:inorder(root);
           break;
           case 3: exit(0);
           default: printf("\nInvalid Choice\n");
        }
        }
    return 0;
}
```