```
#include<iostream>
#include<iomanip>
#include<stdlib.h>
#include<time.h>
#include<windows.h>
#include<graphics.h>

using namespace std;

struct node
{
    int data;
    char colour;
    node *left,*right,*parent;
};

node *root=NULL;

void bothred(node *g,node *u,node *p)
{
    if(g!=root)
        g->colour='r';
    u->colour='b';
    p->colour='b';
}

void uncleblack(node *g,node *u,node *p,node *c)
{
         if(((g->left==u)||(g->left==NULL))&&p->left==c)
         {
             g->right=c;
             c->parent=g;
             p->left=c->right;
             c->right=p;
             p->parent=c;
             swap(p,c);
         }
         if(((g->right==u)||(g->right==NULL))&&p->right==c)
         {
             g->left=c;
             c->parent=g;
             p->right=c->left;
             c->left=p;
             p->parent=c;
             swap(p,c);
         }
         if(g->right==p&&p->right==c)
         {
             g->right=p->left;
             node *sumner=p->left;
             if(sumner!= NULL)
                 sumner->parent=g;
             p->left=g;
             p->parent=g->parent;
             if(g!=root)
             {
                 if((g->parent)->right==g)
                    (g->parent)->right=p;
                 if((g->parent)->left==g)
                    (g->parent)->left=p;
             }
             g->parent=p;
             if(g==root)
                root=p;
             swap(p->colour,g->colour);
             swap(p,g);
         }
         if(g->left==p&&p->left==c)
         {
             g->left=p->right;
             node *sumne=p->right;
             if(sumne != NULL)
                 sumne->parent=g;
             p->right=g;
             p->parent=g->parent;
             if(g!=root)
             {
                 if((g->parent)->right==g)
                    (g->parent)->right=p;
                 if((g->parent)->left==g)
                    (g->parent)->left=p;
             }
             g->parent=p;
             if(g==root)
```

```
85                  root=p;
86                swap(p->colour,g->colour);
87                swap(p,g);
88             }
89  }
90  void checkfunction(node *child)
91  {
92      node *gramps=NULL,*par=NULL,*uncle=NULL;
93      par=child->parent;
94      if(par==NULL)
95          return;
96      gramps=(par)->parent;
97      if(gramps!=NULL)
98      {
99          if(gramps->left==par)
100             uncle=(gramps)->right;
101         else
102             uncle=(gramps)->left;
103         if(uncle!=NULL)
104         {
105             if(uncle->colour=='r'&&par->colour=='r'&&child->colour=='r')
106                 bothred(gramps,uncle,par);
107             if(uncle->colour=='b'&&par->colour=='r'&&child->colour=='r')
108                 uncleblack(gramps,uncle,par,child);
109         }
110         if(uncle==NULL)
111             uncleblack(gramps,uncle,par,child);
112     }
113 }
114 void preorder(node *ptr)
115 {
116     node *temp=new node;
117     temp=ptr;
118     if(temp!=NULL)
119     {
120         checkfunction(temp);
121         preorder(temp->left);
122         preorder(temp->right);
123     }
124 }
125 void inorder(node *ptr)
126 {
127     node *temp=new node;
128     temp=ptr;
129     if(temp!=NULL)
130     {
131         inorder(temp->left);
132         cout<<"\t\t\t\t\t\t\t\t\t                    ";
133         cout<<left<<setw(10)<<temp->data<<left<<setw(10)<<temp->colour;
134         if(temp!=root)
135         {
136             cout<<left<<setw(10)<<(temp->parent)->data<<endl;
137         }
138         else
139             cout<<endl;
140         inorder(temp->right);
141     }
142 }
143 void insertnode()
144 {
145     node*par=NULL;
146     node* newnode=new node;
147     cout<<"\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\t\t\t\t\t\t\t                    Enter
    the value of new node to be inserted:";
148     cin>>newnode->data;
149     newnode->left=NULL;
150     newnode->right=NULL;
151     newnode->parent=NULL;
152     newnode->colour='r';
153     if(root==NULL)
154     {
155         root=newnode;
156         root->colour='b';
157         cout<<"\n\n\n\t\t\t\t\t\t                    Root value : "<<newnode->data<<endl;
158         return;
159     }
160     else
161     {
162         node *cur=root;
163         while(cur!=NULL)
164         {
165             par=cur;
166             if(newnode->data>cur->data)
167                 cur=cur->right;
```

```
168             else
169                 if(newnode->data<cur->data)
170                     cur=cur->left;
171                 else
172                 {
173                     cout<<"\n\n\n\t\t\t\t\t\t\t                    Node "<<newnode->data<<"
    already present.."<<endl;
174                     return;
175                 }
176         }
177         if(newnode->data>par->data)
178         {
179             par->right=newnode;
180             newnode->parent=par;
181         }
182         else
183         {
184             par->left=newnode;
185             newnode->parent=par;
186         }
187         checkfunction(newnode);
188         preorder(root);
189         cout<<"\n\n\n\t\t\t\t\t\t\t                    Node "<<newnode->data<<" successfully
    inserted..."<<endl;
190     }
191     return;
192 }
193 void eliminatedb(node *elnode);
194 void dpreorder(node *ptr)
195 {
196     node *temp=new node;
197     temp=ptr;
198     if(temp!=NULL)
199     {
200         if(temp->colour=='d')
201             eliminatedb(temp);
202         dpreorder(temp->left);
203         dpreorder(temp->right);
204     }
205 }
206 void siblingred(char pos,node *elnode)
207 {
208         node *db=elnode;
209         node *g=db->parent;
210         node *p=NULL,*pl=NULL,*pt=NULL;
211         if(db==g->left)
212             p=g->right;
213         else
214             if(db==g->right)
215                 p=g->left;
216         swap(p->colour,g->colour);
217         if(pos=='r')
218         {
219             pl=p->right;
220             pt=p->left;
221             g->right=p->left;
222             node *sumner=p->left;
223             if(sumner!= NULL)
224                 sumner->parent=g;
225             p->left=g;
226             p->parent=g->parent;
227             if(g!=root)
228             {
229                 if((g->parent)->right==g)
230                     (g->parent)->right=p;
231                 if((g->parent)->left==g)
232                     (g->parent)->left=p;
233             }
234             g->parent=p;
235             if(g==root)
236                 root=p;
237             dpreorder(root);
238         }
239         if(pos=='l')
240         {
241             pl=p->left;
242             pt=p->right;
243             g->left=p->right;
244             node *sumne=p->right;
245             if(sumne != NULL)
246                 sumne->parent=g;
247             p->right=g;
248             p->parent=g->parent;
249             if(g!=root)
```

```
250                 {
251                     if((g->parent)->right==g)
252                         (g->parent)->right=p;
253                     if((g->parent)->left==g)
254                         (g->parent)->left=p;
255                 }
256                 g->parent=p;
257                 if(g==root)
258                     root=p;
259                 dpreorder(root);
260             }
261     }
262     void linered(char pos,node *elnode)
263     {
264         node *db=elnode;
265         node *g=db->parent;
266         node *p=NULL,*pl=NULL,*pt=NULL;
267         if(db==g->left)
268             p=g->right;
269         else
270             if(db==g->right)
271                 p=g->left;
272         swap(g->colour,p->colour);
273         if(pos=='r')
274         {
275             pl=p->right;
276             pt=p->left;
277             g->right=p->left;
278             node *sumner=p->left;
279             if(sumner!= NULL)
280                 sumner->parent=g;
281             p->left=g;
282             p->parent=g->parent;
283             if(g!=root)
284             {
285                 if((g->parent)->right==g)
286                     (g->parent)->right=p;
287                 if((g->parent)->left==g)
288                     (g->parent)->left=p;
289             }
290             g->parent=p;
291             if(g==root)
292                 root=p;
293             if(db->data==-1111)
294             {
295                 if((db->parent)->left==db)
296                     (db->parent)->left=NULL;
297                 else
298                     (db->parent)->right=NULL;
299                 delete(db);
300             }
301         }
302         if(pos=='l')
303         {
304             pl=p->left;
305             pt=p->right;
306             g->left=p->right;
307             node *sumne=p->right;
308             if(sumne != NULL)
309                 sumne->parent=g;
310             p->right=g;
311             p->parent=g->parent;
312             if(g!=root)
313             {
314                 if((g->parent)->right==g)
315                     (g->parent)->right=p;
316                 if((g->parent)->left==g)
317                     (g->parent)->left=p;
318             }
319             g->parent=p;
320             if(g==root)
321                 root=p;
322             if(db->data==-1111)
323             {
324                 if((db->parent)->left==db)
325                     (db->parent)->left=NULL;
326                 else
327                     (db->parent)->right=NULL;
328                 delete(db);
329             }
330         }
331         pl->colour='b';
332     }
333     void trianglered(char pos,node *elnode)
```

```
{
    node *db=elnode;
    node *g=db->parent;
    node *p=NULL,*pl=NULL,*pt=NULL;
    if(db==g->left)
        p=g->right;
    else
        if(db==g->right)
            p=g->left;
    if(pos=='r')
    {
        pl=p->right;
        pt=p->left;
        g->right=pt;
        pt->parent=g;
        p->left=pt->right;
        pt->right=p;
        p->parent=pt;
    }
    if(pos=='l')
    {
        pl=p->left;
        pt=p->right;
        g->left=pt;
        pt->parent=g;
        p->right=pt->left;
        pt->left=p;
        p->parent=pt;
    }
    swap(p->colour,pt->colour);
    linered(pos,elnode);
}
void eliminatedb(node *elnode)
{
    node *db=elnode;
    if(db==root)
    {
        db->colour='b';
        return;
    }
    node *g=db->parent;
    node *p=NULL,*pl=NULL,*pt=NULL;
    char flag;
    if(db==g->left)
    {
        p=g->right;
        flag='r';
        if(p!=NULL)
        {
            pl=p->right;
            pt=p->left;
        }
    }
    else
        if(db==g->right)
        {
            p=g->left;
            flag='l';
            if(p!=NULL)
            {
                pl=p->left;
                pt=p->right;
            }
        }
    if(p!=NULL)
    {
        if(p->colour=='r')
        {
            siblingred(flag,db);
        }
        else if(p->colour=='b')
        {
            if(pt!=NULL)
            {
                if(pt->colour=='r')
                {
                    trianglered(flag,db);
                }
            }
            else if(pl!=NULL)
            {
                if(pl->colour=='r')
                {
                    linered(flag,db);
```

```
                }
            }
            else
if((pl==NULL&&pt==NULL)||(pl->colour=='b'&&pt->colour=='b')||(pl==NULL&&pt->colour=='b')||(pt
==NULL&&pl->colour=='b'))
            {
                db->colour='b';
                p->colour='r';
                if(g->colour=='r')
                    g->colour='b';
                else if(g->colour=='b')
                    g->colour='d';
                if(db->data==-1111)
                {
                    if((db->parent)->left==db)
                        (db->parent)->left=NULL;
                    else
                        (db->parent)->right=NULL;
                    delete(db);
                }
                if(g->colour=='d')
                    eliminatedb(g);
            }
        }
    }
    else if((p==NULL)||(p->colour=='b'&&pl->colour=='b'&&pt->colour=='b'))
    {
        if(flag=='r')
            g->left=NULL;
        else
            if(flag=='l')
                g->right=NULL;
        db->colour='b';
        if(g->colour=='r')
            g->colour='b';
        else if(g->colour=='b')
            g->colour='d';
        if(p!=NULL)
            p->colour='r';
        if(db->data==-1111)
        {
            if((db->parent)->left==db)
                (db->parent)->left=NULL;
            else
                (db->parent)->right=NULL;
            delete(db);
        }
        if(g->colour=='d')
            eliminatedb(g);
    }
}
void findpredorsuc(node *del)
{
    node *predecessor=NULL,*successor=NULL,*temp=NULL;
    if(del->left==NULL)
    {
        if(del->right!=NULL)
            successor=del->right;
        else
            {
                if(del!=root)
                {
                    if(del->colour=='b')
                    {
                        del->colour='d';
                        temp=del;
                        temp->data=-1111;//eyes on this.
                        eliminatedb(temp);
                    }
                    else if(del->colour=='r')
                    {
                        if((del->parent)->left==del)
                            (del->parent)->left=NULL;
                        else
                            (del->parent)->right=NULL;
                        delete(del);
                    }
                }
                else if(del==root)
                    delete(del);
            }
    }
    else
        predecessor=del->left;
```

```
500         if(predecessor!=NULL)
501         {
502             while(predecessor!=NULL)
503             {
504                 temp=predecessor;
505                 predecessor=predecessor->right;
506 
507             }
508             swap(temp->data,del->data);
509             if(temp->colour=='r')
510             {
511                 if((temp->parent)->left==temp)
512                     (temp->parent)->left=NULL;
513                 else
514                     (temp->parent)->right=NULL;
515                 delete(temp);
516             }
517             if(temp->colour=='b')
518             {
519                 temp->colour='d';
520                 temp->data=-1111;
521                 eliminatedb(temp);
522             }
523         }
524         if(successor!=NULL)
525         {
526             while(successor!=NULL)
527             {
528                 temp=successor;
529                 successor=successor->left;
530             }
531             swap(temp->data,del->data);
532             if(temp->colour=='r')
533             {
534                 if((temp->parent)->left==temp)
535                     (temp->parent)->left=NULL;
536                 else
537                     (temp->parent)->right=NULL;
538                  delete(temp);
539             }
540             if(temp->colour=='b')
541             {
542                 temp->colour='d';
543                 temp->data=-1111;
544                 eliminatedb(temp);
545             }
546         }
547     }
548     void deletenode()
549     {
550         int dn;
551 
    cout<<"\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\t\t\t\t\t\t\t\t\t\t
    Enter the node to be deleted:";
552         cin>>dn;
553         node *temp=root;
554         while(temp!=NULL)
555         {
556             if(dn==temp->data)
557                 break;
558             else
559                 if(dn>temp->data)
560                     temp=temp->right;
561                 else
562                     if(dn<temp->data)
563                         temp=temp->left;
564 
565         }
566         if(temp!=NULL)
567         {
568             findpredorsuc(temp);
569             cout<<"\n\n\n\t\t\t\t\t\t\t\t\t                    Node "<<dn<<" deleted
    successfully"<<endl;
570             if(temp==root&&temp->left==NULL&&temp->right==NULL)
571                 root=NULL;
572         }
573         else
574         {
575             cout<<"Node not present...";
576             return;
577         }
578     }
579     int main()
580     {
```

```
581         system("cls");
582         system("COLOR 04");
583         cout<<"\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\t\t\t\t\t\t\t\t\t\t
    ------------------------------------------"<<endl;
584         cout<<"\t\t\t\t\t\t\t\t\t     |ADA Open Ended Project on RED-BLACK TREES|"<<endl;
585         cout<<"\t\t\t\t\t\t\t\t\t     ------------------------------------------"<<endl;
586         delay(10000);
587         system("cls");
588         int choice=1;
589         int chc;
590         while(choice)
591         {
592             system("cls");
593             system("COLOR 04");
594             cout<<"\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\t\t\t\t\t\t\t\t\t
    1.INSERT\n\n\t\t\t\t\t\t\t\t\t\t\t
    2.DELETE\n\n\t\t\t\t\t\t\t\t\t\t                    3.DISPLAY\n\n\n\t\t\t\t\t\t\t Enter your
    choice:";
595             cin>>chc;
596             if(chc==1)
597             {
598                 system("cls");
599                 insertnode();
600             }
601             if(chc==2)
602             {
603                  system("cls");
604                 deletenode();
605             }
606             if(chc==3)
607             {
608                  system("cls");
609
    cout<<"\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\t\t\t\t\t\t\t                    ";
610                 if(root==NULL)
611                    cout<<"Empty tree...."<<endl;
612                 else
613                 {
614
    cout<<left<<"\t"<<setw(10)<<"Value"<<left<<setw(10)<<"Colour"<<left<<setw(10)<<"Parent"<<endl
    <<endl;
615                    inorder(root);
616                 }
617             }
618             cout<<"\n\n\n\t\t\t\t\t\t\t\t\t                  Do you want to continue:";
619             cin>>choice;
620             system("cls");
621         }
622         cout<<"\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\t\t\t\t\t\t\t
    Thank you for using this code"<<endl;
623         return 0;
624     }
625
```