/* Adelson Velsekii and Landis Tree with Graphics Display */

/*
Polytechnic University of the Philippines
Mabini Campus, Sta. Mesa, Manila
College of Computer Management and Information Technology

A Partial Fulfillment of the Subject Design and Analysis of Algorithm

Programmed by:
Frederick A. Badion
BS Computer Science 2nd year (SY: 2K5-2K6)

Subject Professor:
Mrs. Ria Sagum
Chaiperson BSCompSci

*/

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <dos.h>
#include <string.h>
#include <graphics.h>
#include <conio.h>

struct node
{
  int element;
  node *left;
  node *right;
  int height;
};

        typedef struct node *nodeptr;

        void insert(int,nodeptr &);
        void del(int, nodeptr &);
        int deletemin(nodeptr &);
        void find(int,nodeptr &);
        nodeptr findmin(nodeptr);
        nodeptr findmax(nodeptr);
        void makeempty(nodeptr &);
        nodeptr nodecopy(nodeptr &);
        void preorder(nodeptr);
        void inorder(nodeptr);
        void postorder(nodeptr);
        int bsheight(nodeptr);
        nodeptr singlerotateleft(nodeptr &);
        nodeptr doublerotateleft(nodeptr &);
        nodeptr singlerotateright(nodeptr &);
        nodeptr doublerotateright(nodeptr &);
        int max(int,int);
        int nonodes(nodeptr);


int gdriver=DETECT,gmode=0,errorcode;
char element[3];
int x=1,maxx,midx,xcoord,ycoord,level=320,prevlevel;

void GDisplay(nodeptr p,int xcoord,int ycoord)
{
 if (p!=NULL)
        {
          level=level/2;
          setfillstyle(1,BROWN);
          setcolor(LIGHTGREEN);
          if(p->left->element!=NULL)
                line(xcoord,ycoord,xcoord-level,ycoord+50);
          if(p->right->element!=NULL)
                line(xcoord,ycoord,xcoord+level,ycoord+50);
          fillellipse(xcoord,ycoord,10,10);
          sprintf(element,"%d",p->element,xcoord,ycoord);
          settextstyle(2,0,4);
          setcolor(YELLOW);
          outtextxy(xcoord-7,ycoord-7,element);
          GDisplay(p->left,xcoord-(level),ycoord+50);
          GDisplay(p->right,xcoord+(level),ycoord+50);
```

```
            level=level*2;
        }
}//end of GDisplay


void insert(int x,nodeptr &p)
{
  if (p == NULL)
  {
        p = new node;
        p->element = x;
        p->left=NULL;
        p->right = NULL;
        p->height=0;
        if (p==NULL)
            {
                gotoxy(4,21);
                printf("Out of Space");
            }
  }
  else
  {
        if (x<p->element)
        {
          insert(x,p->left);
          //GDisplay(root,midx,50);
  if ((bsheight(p->left) - bsheight(p->right))==2)
  {
    if (x < p->left->element)
        p = singlerotateleft(p);    //single rotation left
    else
        p = doublerotateleft(p);  //double rotation left
  }
}
else if (x>p->element)
{
    insert(x,p->right);
    //GDisplay(root,midx,50);
    if ((bsheight(p->right) - bsheight(p->left))==2)
    {
        if (x > p->right->element)
                p = singlerotateright(p);  //single rotation right
        else
                p = doublerotateright(p); //double rotation right
    }
}
else
   {
        gotoxy(4,21);
        printf("Element Exists");
   }
}
int m,n,d;
m=bsheight(p->left);
n=bsheight(p->right);
        d=max(m,n);
        p->height = d + 1;
}

nodeptr findmin(nodeptr p)
{
        if (p==NULL)
        {
           gotoxy(4,21); printf("Empty Tree");
           return p;
        }
        else
        {
          while(p->left !=NULL)
                p=p->left;
          return p;
        }
}

nodeptr findmax(nodeptr p)
{
        if (p==NULL)
        {
          gotoxy(4,21); printf("Empty Tree");
          return p;
        }
```

```
        else
        {
          while(p->right !=NULL)
             p=p->right;
          return p;
        }
}

void find(int x,nodeptr &p)
{
        if (p==NULL)
        {
          gotoxy(4,21);
          printf("Element not found");
        }
        else if (x < p->element)
          find(x,p->left);
        else if (x>p->element)
          find(x,p->right);
        else
        {
          gotoxy(4,21);
          printf("Element found !");
        }
}
```

```
void del(int x,nodeptr &p)
{
        nodeptr d;
        if (p==NULL)
        {
          gotoxy(4,21);
          printf("Element not found");
        }
        else if ( x < p->element)
        {
          del(x,p->left);
          if ((bsheight(p->left) - bsheight(p->right))==2)
          {
            if (x < p->left->element)
                p = singlerotateleft(p);    //single rotation left
            else
                p = doublerotateleft(p);  //double rotation left
          }
        }
        else if (x > p->element)
        {
          del(x,p->right);
          if ((bsheight(p->right) - bsheight(p->left))==2)
          {
            if (x > p->right->element)
                p = singlerotateright(p);  //single rotation right
            else
                p = doublerotateright(p); //double rotation right
          }
        }
        else if ((p->left == NULL) && (p->right == NULL))
        {
           d=p;
           free(d);
           p=NULL;
           gotoxy(4,21); printf("Element deleted !");
        }
        else if (p->left == NULL)
        {
          d=p;
          free(d);
          p=p->right;
          gotoxy(4,21); printf("Element deleted !");
        }
        else if (p->right == NULL)
        {
          d=p;
          p=p->left;
          free(d);
          gotoxy(4,21); printf("Element deleted !");
        }
        else
          p->element = deletemin(p->right);
}
```

```
int deletemin(nodeptr &p)
{
        int c;
        gotoxy(4,21); printf("deltemin");
        if (p->left == NULL)
        {
          c=p->element;
          p=p->right;
          return c;
        }
        else
        {
          c=deletemin(p->left);
          return c;
        }
}

void preorder(nodeptr p)
{
        if (p!=NULL)
        {
          printf("%d-->",p->element);
          preorder(p->left);
          preorder(p->right);
        }
}

void inorder(nodeptr p)
{
        if (p!=NULL)
        {
          inorder(p->left);
          printf("%d-->",p->element);
          inorder(p->right);
        }
}

void postorder(nodeptr p)
{
        if (p!=NULL)
        {
          postorder(p->left);
          postorder(p->right);
          printf("%d-->",p->element);
        }
}

int max(int value1, int value2)
{
        if(value1 > value2)
          return value1;
        else
          return value2;
}

int bsheight(nodeptr p)
{
        int t;
        if (p == NULL)
                return -1;
        else
        {
                t = p->height;
                return t;
        }
}

nodeptr singlerotateleft(nodeptr &p1)
{
        nodeptr p2;
        p2 = p1->left;
        p1->left = p2->right;
        p2->right = p1;
        p1->height = max(bsheight(p1->left),bsheight(p1->right)) + 1;
        p2->height = max(bsheight(p2->left),p1->height) + 1;
        return p2;
}

nodeptr singlerotateright(nodeptr &p1)
{
        nodeptr p2;
```

```
        p2 = p1->right;
        p1->right = p2->left;
        p2->left = p1;
        p1->height = max(bsheight(p1->left),bsheight(p1->right)) + 1;
        p2->height = max(p1->height,bsheight(p2->right)) + 1;
        return p2;
}

nodeptr doublerotateleft(nodeptr &p1)
{
        p1->left = singlerotateright(p1->left);
        return singlerotateleft(p1);
}

nodeptr doublerotateright(nodeptr &p1)
{
        p1->right = singlerotateleft(p1->right);
        return singlerotateright(p1);
}

int count=0;
int nonodes(nodeptr p)
{
        if (p!=NULL)
        {
                nonodes(p->left);
                nonodes(p->right);
                count++;
        }
        return count;
}

void twolinebox(int x1,int y1,int x2,int y2){
 int x,y;
 textcolor(WHITE);
 gotoxy(x1,y1); cprintf("É"); //201
 gotoxy(x2,y1); cprintf("»"); //187
  for(y=y1+1;y<y2;y++){
    gotoxy(x1,y); cprintf("º"); //186
    gotoxy(x2,y); cprintf("º"); //186
  }
 gotoxy(x1,y2); cprintf("È"); //200
 gotoxy(x2,y2); cprintf("¼"); //188
  for(x=x1+1;x<x2;x++){
    gotoxy(x,y1); cprintf("Í"); //205
    gotoxy(x,y2); cprintf("Í"); //205
  }
 gotoxy(x1+1,y1+1);
}

void cprintxy(int x,int y,int color,char string[]){
 textcolor(color);
 gotoxy(x,y); cprintf("%s",string);
}

void center(int y,int color,char string[]){
int x=(80-strlen(string)+1)/2;
textcolor(color);
gotoxy(x,y);cprintf("%s",string);
}


void Temp(void){
int x,y;
clrscr();
twolinebox(1,1,80,24);
for(y=23;y>=1;y--){
        sound(60*y);
        center(y,YELLOW,"[Adelson-Velskii and Landis Tree]");
        gotoxy(2,y+1); clreol();
        twolinebox(1,1,80,24);
        delay(40);
        nosound();
}
center(1,YELLOW,"[Adelson-Velskii and Landis Tree]");
for(x=74;x>=3;x--){
        sound(50*x);
        cprintxy(x,5,RED,"Press:");   clreol();
        twolinebox(1,1,80,24);
        center(1,YELLOW,"[Adelson-Velskii and Landis Tree]");
```

```
		delay(20);
		nosound();
}
twolinebox(1,1,80,12);
twolinebox(1,1,80,24);
center(1,YELLOW,"[Adelson-Velskii and Landis Tree]");
cprintxy(20,3,GREEN,"[A]- Insertion");
cprintxy(20,4,GREEN,"[B]- Find Minimum");
cprintxy(20,5,GREEN,"[C]- Find Maximum");
cprintxy(20,6,GREEN,"[D]- Search Node");
cprintxy(20,7,GREEN,"[E]- Display Tree");
cprintxy(43,3,GREEN,"[F]- Delete Node");
cprintxy(43,4,GREEN,"[G]- Preorder");
cprintxy(43,5,GREEN,"[H]- Inorder");
cprintxy(43,6,GREEN,"[I]- Postorder");
cprintxy(43,7,GREEN,"[J]- Height");
cprintxy(20,9,GREEN,"[any key]- Quit...");

cprintxy(20,11,LIGHTRED,"Enter your choice:    ");
}

void main()
{

		nodeptr root,min,max;
		int a,findele,delele,leftele,rightele,flag;
char choice,value[2];
char ch='Y';
root = NULL;
textmode(C80);
Temp();
do
{
		clrscr();
		twolinebox(1,1,80,12);
		twolinebox(1,1,80,24);
		center(1,YELLOW,"[Adelson-Velskii and Landis Tree]");
		cprintxy(5,3,LIGHTRED,"Press: ");
		cprintxy(20,3,GREEN,"[A]- Insertion");
		cprintxy(20,4,GREEN,"[B]- Find Minimum");
		cprintxy(20,5,GREEN,"[C]- Find Maximum");
		cprintxy(20,6,GREEN,"[D]- Search Node");
		cprintxy(20,7,GREEN,"[E]- Display Tree");
		cprintxy(43,3,GREEN,"[F]- Delete Node");
		cprintxy(43,4,GREEN,"[G]- Preorder");
		cprintxy(43,5,GREEN,"[H]- Inorder");
		cprintxy(43,6,GREEN,"[I]- Postorder");
		cprintxy(43,7,GREEN,"[J]- Height");
		cprintxy(20,9,GREEN,"[any key]- Quit...");

		cprintxy(20,11,LIGHTRED,"Enter your choice:\>");
		choice=getch();
		switch(toupper(choice))
		{
		case 'A':
			do{

			gotoxy(4,14); printf("Enter node value: ");
			a=atoi(gets(value));
			if(atoi(value)==0)
			{
				gotoxy(4,21); printf("Error!!! Enter a valid integer value only.");
				gotoxy(4,22); printf("Press any key to continue...");
				getch();
			}
			}while(atoi(value)==0);
			insert(a,root);
			gotoxy(4,15);
			inorder(root);
			/*
			initgraph(&gdriver,&gmode,"c:\tc\bgi");
			errorcode = graphresult();
			maxx=getmaxx();
			midx=maxx/2,xcoord=midx/2,ycoord=40;
			if (errorcode != grOk)
			 {
			   printf("Graphics error: %s
", grapherrormsg(errorcode));
			   printf("Press any key to stop");
			   getch();
			   exit(1);
```

```
            }
            cleardevice();
            GDisplay(root,midx,50);
            getch();
            restorecrtmode();
            */
           break;
case 'B':
           if (root !=NULL)
           {
           min=findmin(root);
           gotoxy(4,14); printf("Min element :  %d",min->element);
           }
           break;
 case 'C':
           if (root !=NULL)
           {
           max=findmax(root);
           gotoxy(4,14); printf("Max element :  %d",max->element);
           }
           break;
 case 'D':
           gotoxy(4,14); printf("Search node :");
           scanf("%d",&findele);
           if (root != NULL)
                    find(findele,root);
                    break;
                case 'E':
                    initgraph(&gdriver,&gmode,"c:\tc\bgi");
                    errorcode = graphresult();
                    maxx=getmaxx();
                    midx=maxx/2;
                    xcoord=midx/2;
                    ycoord=40;

                    if (errorcode != grOk)
                     {
                      printf("Graphics error: %s
", grapherrormsg(errorcode));
                      printf("Press any key to stop");
                      getch();
                      exit(1);
                     }
                     cleardevice();
                     setbkcolor(LIGHTBLUE);
                     settextstyle(2,0,5);
                     outtextxy(20,10,"Adelson-Velskii and Landis Tree");
                     GDisplay(root,midx,50);
                     outtextxy(20,440,"Programmed by: Frederick Badion");
                     outtextxy(20,450,"Polytechnic University of the Philippines");
                     outtextxy(20,460,"2nd year Bachelor of Science in Computer
Science");
                     outtextxy(320,440,"A partial fulfilment for the subject: ");
                     outtextxy(320,450,"Design and Analysis of Algorithm");
                     outtextxy(320,460,"Prof. Ria Sagum -Chairperson BSCS-CCMIT
PUP-Sta.Mesa");
                      getch();
                      restorecrtmode();
                     break;
                case 'F':
                     gotoxy(4,14); printf("Delete Node: ");
                     scanf("%d",&delele);
                     del(delele,root);
                     getch();
                     initgraph(&gdriver,&gmode,"c:\tc\bgi");
                     errorcode = graphresult();
                     maxx=getmaxx();
                     midx=maxx/2,xcoord=midx/2,ycoord=40;
                     if (errorcode != grOk)
                      {
                       printf("Graphics error: %s
", grapherrormsg(errorcode));
                       printf("Press any key to stop");
                       getch();
                       exit(1);
                      }
                     cleardevice();
                     setbkcolor(LIGHTBLUE);
                     settextstyle(2,0,5);
                     outtextxy(20,10,"Adelson-Velskii and Landis Tree");
                     GDisplay(root,midx,50);
```

```
                getch();
                restorecrtmode();
                break;

        case 'G':
                gotoxy(4,14); printf(" Preorder Printing...");
                gotoxy(4,15);
                preorder(root);
                break;

        case 'H':
                gotoxy(4,14); printf(" Inorder Printing...");
                gotoxy(4,15);
                inorder(root);
                break;

        case 'I':
                gotoxy(4,14); printf(" Postorder Printing...");
                gotoxy(4,15);
                postorder(root);
                break;
        case 'J':
                gotoxy(4,14); printf(" Height and Depth: %d",bsheight(root));
                gotoxy(4,15); printf("No. of nodes: %d",nonodes(root));
                break;
        }
                gotoxy(4,22); printf(" Do you want to continue (y/n)?");
                ch=toupper(getch());
        }while(ch!='N');
}
```