Government Engineering College, Gandhinagar

Computer Engineering

B.E. Semester III
(AY 2021-22)

Student Details:

Enrolment No. :	200130107044
Full Name :	Prajapati Jay Bharatbhai
Sub-Division:	A2

Signatures:

Student:	
Subject Coordinator/Lab Faculty:	
HOD Signature:	

Index

Brief idea about working of the project

The project:

- ► Has been made in C++.
- ► Uses animation to describe working Binary Search Tree
- ► Takes input from the user and produces an animation accordingly.

The codes to this project can be found at:

https://github.com/Jay2219/Binary-Search-Tree-GUI

Video of Implementation of Code:

https://drive.google.com/file/d/1s5yWzo5TLFrZ8a3cb0yNm8UYM3Vnf94/view

Roadblocks faced while making the project

- The graphics.h library has simple functions which are easy to use and grasp.
- The issue was that graphics.h being an almost 20 year old library is not available in the latest versions of devcpp, codeblocks or other IDEs.
- A lot of ways were present online to make graphics.h work in several versions of several different IDEs. However many of them were themselves outdated and many of the rest didn't work properly.
- It took quite some time and effort to find a way which actually worked in Turbo C++ 3.7.8.9 on Windows 10.

Various graphics.h commands

(I) Used for creating the graphics window

- initgraph(&gdriver,&gmode,"C:\\Turboc3\\BGI");

1) It initializes the graphic system and loads the graphic drivers. It then puts the system in graphics mode.

2) &gdriver is an integer that specifies which graphics driver to be used.

3) &gmode specifies the initial graph mode. If graphmode is set to DETECT then initgraph sets &gmode to the highest resolution available for the detected driver.

- **closegraph()** : It closes the graph.

(II) Used for drawing shapes and animating

- **drawLine(int X0,int Y0,int X1,int Y1)** : Draws a line from the point (X0,Y0) in the coordinate system to the point (X1,Y1).
- **drawCircle(int xc,int yc,int x,int y)** : Put pixels of circle at (x,y) with center (xc,yc).
- **circleBres(int xc,int yc,int r)** : Join the circle's pixels with center (xc,yc) at radius r.
- **drawTree(struct node*ptr,char LorR,int depth,int xc,int yc,int xc1,int yc1) :** Draw the node of tree at its valid position
- **outtextxy(int x, int y, str)** : Displays the text stored in str at (x,y)
- **setcolor(<colorname>)** : Sets the color which the rest of the objects drawn on the graph appear in.
- **setbkcolor(<colorname>)** : Sets the background color.

- **delay(<time in milliseconds>)** : Causes a delay inexecution by the entered number of milliseconds.
- **cleardevice()** : Clears the graph

Basic idea behind animation

- The basic working behind the animation is that we can create an object, remove it repeatedly, in a loop so as to make it seem as if the object is moving.
- The speed of the object can be controlled using the delay() function. More the delay, less the speed of the object.

Binary Search Tree

Binary Search Tree is a node-based binary tree data structure which has the following properties:

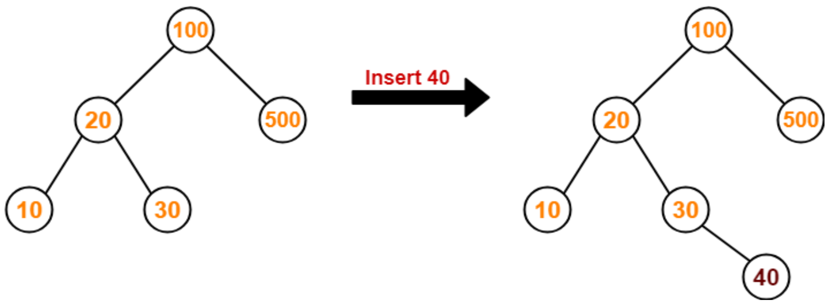
- ► All the nodes of the left subtree are lesser than the node itself.
- ► All the nodes of the right subtree are greater than the node itself.
- ► The left and right both subtrees are each must be a 'Binary Search Tree'.
- ► There must be no duplicate nodes.

Basic operations in Binary Search Tree:

► **Insert()** :

1) This operation inserts an element always at the leaf.

2) We would simply start from the root node and see that the element that we want to insert is greater or lesser than the node.



3)Since, here 40 is less than 100 we move to the left root. And then the root is 20, and since there is 40 greater than 20, we move to the right root. And then root is element 30 and since there is 40 greater than 30, we move to right root.And since there are no elements to the right, we simply insert element 40 there

► **DELETE()**:

- In case of deletion there are three cases:

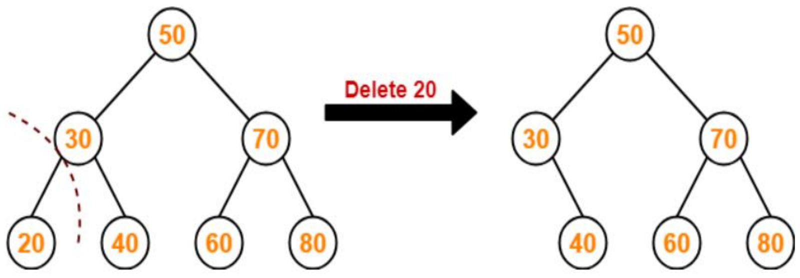
  (1) The node is a leaf node.

  (2) The node is a non-leaf node

  (3) The node is the root node

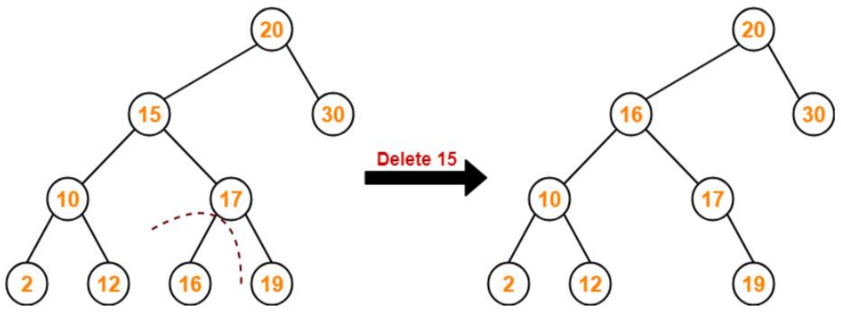
(1) Deleting leaf node :

- In this case of deletion in Binary Search Tree, first we have to search the element that we have to delete and then remove it from the Tree. And at last make its parent node point to NULL.

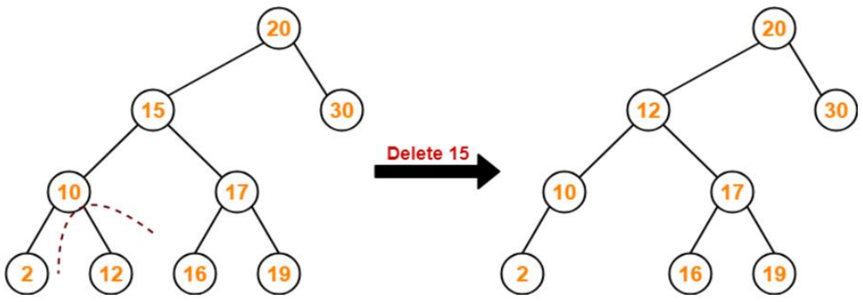


(2) Deleting non-leaf node :

- In this case of deletion in Binary Search Tree, first we have to search the element that we have to delete.
- And then we have to replace that node by an element which is InOrder Predecessor or InOrder Successor in InOrder Traversal.



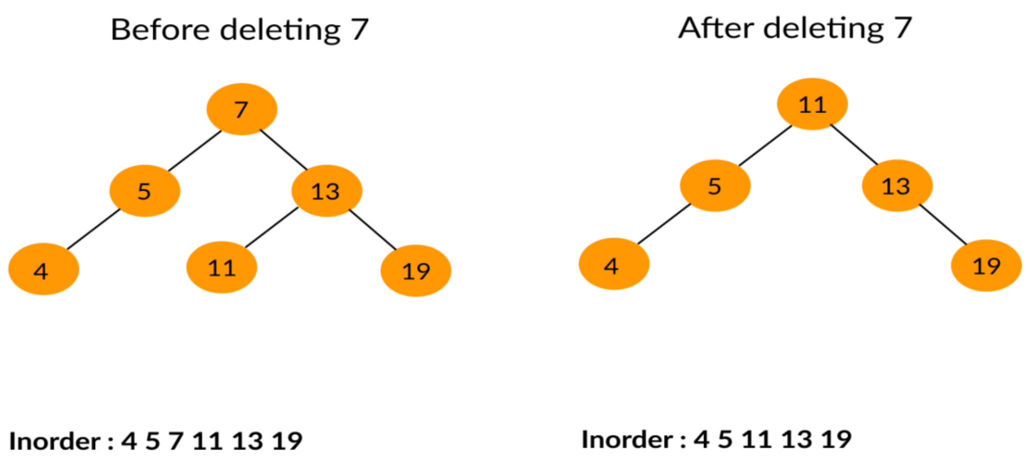
- Here we have to delete 15 and the InOrder Traversal is 2→10→12→**<u>15</u>**→16→17→19→20→30. Here InOrder Successor is 16. Above figure shows Deletion by replacing Successor.



- Here the we have to delete 15 and the InOrder Traversal is 2→10→12→**<u>15</u>**→16→17→19→20→30. Here InOrder Predecessor is 12 . Above figure shows Deletion by replacing Predecessor.

(3) Deleting root node :

- First, search for the node to be deleted
- Search for the InOrder Predecessor and Successor of the node.
- Keep doing that until the tree has no empty nodes.



- Here we first find the element that we want to delete. Then we replace the root node with InOrder predecessor or InOrder successor.

BST Graphic User interface C++ code

```
#include<stdio.h>
#include<malloc.h>
#include<string.h>
#include<graphics.h>
#include<math.h>
#include<conio.h>
#include<dos.h>
#include<stdlib.h>
#define YINC 60
#define RAD 15
#define F getmaxx()
#define G getmaxy()

int midx,anim;
//Nodes of the tree
struct node
{
	int data;
	struct node *left;
	struct node *right;
}*root=NULL;
//Helper function to create new nodes
struct node *newNode(int data)
```

```
{
        struct node *node=(struct node*)malloc(sizeof(struct node));
        node->data=data;
        node->left=NULL;
        node->right=NULL;
        return(node);
}
void displayData(int data,int x,int y)
{
        //Converting integer to string
        int n=data,i=0,j=0;
        char str[10];
        while(n)
        {
                str[i++]=n%10+'0';
                n/=10;
                str[i]='\0';
        }
        for(i--;j<i;j++,i--)
        {
                str[i]+=str[j];
                str[j]=str[i]-str[j];
                str[i]-=str[j];
        }
        setcolor(WHITE);
        outtextxy(x-5,y-5,str);
```

```
	setcolor(RED);
}
void drawLine(int X0,int Y0,int X1,int Y1)
{
	int i;
	//calculate dx & dy
	int dx=X1-X0;
	int dy=Y1-Y0;
	//calculate steps required for generating pixels
	int steps=abs(dx)>abs(dy)?abs(dx):abs(dy);
	//calculateincrementin x&y for each steps
	float Xinc=dx/(float )steps;
	float Yinc=dy/(float )steps; //Putpixelforeachstep
	float X=X0;
	float Y=Y0;
	for(i=0;i<=steps;i++)
	{
		putpixel(X,Y,getcolor());
		X+=Xinc;
		Y+=Yinc;
		if(anim)
		delay(30);
	}
}
void drawCircle(int xc,int yc,int x,int y)
{
        putpixel(xc+x,yc+y,getcolor());
        putpixel(xc-x,yc+y,getcolor());
        putpixel(xc+x,yc-y,getcolor());
        putpixel(xc-x,yc-y,getcolor());
        putpixel(xc+y,yc+x,getcolor());
        putpixel(xc-y,yc+x,getcolor());
        putpixel(xc+y,yc-x,getcolor());
        putpixel(xc-y,yc-x,getcolor());
}
//Functionforcircle-generation //using Bresenham's algorithm
void circleBres(int xc,int yc,int r)
{
        int x=0,y=r;
        int d=3-2*r;
        while(y>=x)
        {
                //foreachpixelwewill //drawalleightpixels
                drawCircle(xc,yc,x,y);
                x++;
                //checkfordecisionparameter
                //andcorrespondingly //updated,x,y
                if(d>0)
                {
                        y--;
                        d=d+4*(x-y)+10;
                }
                else
                d=d+4*x+6;
                drawCircle(xc,yc,x,y);
                if(anim)
                delay(60);
        }
}
void drawTree(struct node*ptr,char LorR,int depth,int xc,int yc,int xc1,int yc1)
{
        if(LorR=='\0')
        circleBres(xc1,yc1,RAD);
        else if(LorR=='L')
        {
                //Calculation for edges
                int len,x_diff,x1,y1,x2,y2;
                x_diff=abs(xc1-xc);
                len=sqrt(pow(x_diff,2)+pow(YINC,2));
                x1=RAD*x_diff/len;
                y1=RAD*YINC/len;
                x2=xc1+x1;
                y2=yc1-y1;
                x1=xc-x1;
                y1=yc+y1;
                drawLine(x1,y1,x2,y2);
                circleBres(xc1,yc1,RAD);
        }
        else
        {
                //Calculationforedges
                int len,x_diff,x1,y1,x2,y2;
                x_diff=abs(xc1-xc);
                len=sqrt(pow(x_diff,2)+pow(YINC,2));
                x1=RAD*x_diff/len;
                y1=RAD*YINC/len;
                x2=xc1-x1;
                y2=yc1-y1;
                x1=xc+x1;
                y1=yc+y1;
                drawLine(x1,y1,x2,y2);
                circleBres(xc1,yc1,RAD);
        }
        //Display data inside the circle
        displayData(ptr->data,xc1,yc1);
}
void calPos(int xc,int yc,int *xc1,int *yc1,int depth,char  LorR)
{
        int i=1,x=midx;
        //Calculatingpositionfornextchild
        for(;(i++)<=depth;x/=2);
        if(xc==-1);
        else if(LorR=='L')
        {
                *xc1=xc-x;
                *yc1=yc+YINC;
        }
        else if(LorR=='R')
        {
                *xc1=xc+x;
                *yc1=yc+YINC;
        }
}
int calTree(struct node*ptr,char  LorR,int depth,int xc,int yc,int xc1,int yc1)
{
        calPos(xc,yc,&xc1,&yc1,depth,LorR);
        if(ptr!=NULL)
        {
                drawTree(ptr,LorR,depth,xc,yc,xc1,yc1);
                calTree(ptr->left,'L',depth+1,xc1,yc1,xc1,yc1);
        }
        else
                return 0;
        calTree(ptr->right,'R',depth+1,xc1,yc1,xc1,yc1);
}
struct node*insert(struct node*ptr,char LorR,int depth,int xc,int yc,int xc1,int
yc1,int data)
{
        calPos(xc,yc,&xc1,&yc1,depth,LorR);
        if(ptr==NULL)
        {
                ptr=newNode(data);
                drawTree(ptr,LorR,depth,xc,yc,xc1,yc1);
                return ptr;
        }
        if(data==ptr->data)
        {
                puts("Duplicate elements are not allowed");
                delay(3000);
                return ptr;
        }
        else  if(data<ptr->data)
                ptr->left=insert(ptr->left,'L',depth+1,xc1,yc1,xc1,yc1,data);
        else
                ptr->right=insert(ptr->right,'R',depth+1,xc1,yc1,xc1,yc1,data);
        return ptr;
}
struct node*minValue(struct node*temp)
{
        while(temp->left!=NULL)
        temp=temp->left;
        return  temp;
}
struct node*free(struct node*ptr,char  LorR,int depth,int xc,int yc,int xc1,int
yc1,int  data)
{
        struct node*temp;
        calPos(xc,yc,&xc1,&yc1,depth,LorR);
```

```
if(ptr==NULL)
{
        printf("Invalid value");
        delay(3000);
        return  ptr;
}
if(data<ptr->data)
        ptr->left=free(ptr->left,'L',depth+1,xc1,yc1,xc1,yc1,data);
else  if(data>ptr->data)
        ptr->right=free(ptr->right,'R',depth+1,xc1,yc1,xc1,yc1,data);
else
{
        setcolor(BLACK);
        if(ptr->left==NULL)
        {
                temp=ptr->right;
                drawTree(ptr,LorR,depth,xc,yc,xc1,yc1);
                free(ptr);
                return  temp;
        }
        else if(ptr->right==NULL)
        {
                temp=ptr->left;
                drawTree(ptr,LorR,depth,xc,yc,xc1,yc1);
                free(ptr);
                return  temp;
                }
                drawTree(ptr,LorR,depth,xc,yc,xc1,yc1);
                temp=minValue(ptr->right);
                ptr->data=temp->data;
                ptr->right=free(ptr->right,'R',depth+1,xc1,yc1,xc1,yc1,temp-
>data);
        }
        return ptr;
}
void boundary(int j,int k);
void main()
{
        int gdriver=0 , gmode;
        int data;
        char cmnd[5];
        clrscr();
        initgraph(&gdriver,&gmode,"C:\\Turboc3\\BGI");
        boundary(15,4);
        cleardevice();
        setbkcolor(1);
        settextstyle(1,0,6);
        outtextxy(130,110,"BINARY SEARCH TREE");
        settextstyle(2,0,5);
        delay(2000);
        cleardevice();
        setbkcolor(6);
        midx=getmaxx()/2;
```

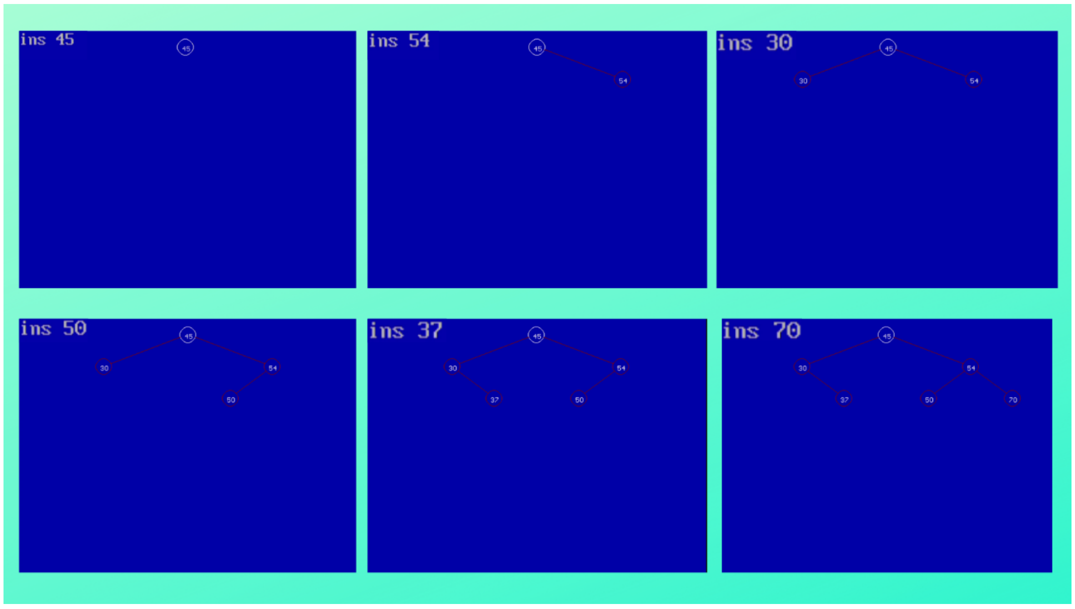
```
while(1)
{
        anim=1;
        scanf("%s",cmnd);
        if(strcmp(cmnd,"ins")==0)
        {
                scanf("%d",&data);
                if(root==NULL)
                        root=insert(root,'\0',0,-1,-1,midx,30,data);
                else
                        insert(root,'\0',0,-1,-1,midx,30,data);
        }
        else  if(strcmp(cmnd,"del")==0)
        {
                scanf("%d",&data);
                if(root==NULL)
                        printf("Tree is already empty\n");
                else
                        root=free(root,'\0',0,-1,-1,midx,30,data);
        }
        else  if(strcmp(cmnd,"root")==0)
        {
                printf("%d\n",root->data);
                delay(3000);
        }
        else  if(strcmp(cmnd,"exit")==0)
```

```
		{
			cleardevice();
			setbkcolor(15);
			boundary(8,5);
			setcolor(5);
			settextstyle(3,0,5);
			outtextxy(130,110,"THANK YOU");
			settextstyle(3,0,4);
			outtextxy(110,180,"FOR ENTERING INTO THE
PORTAL");
			delay(2000);
			closegraph();
			exit(0);
		}
		else
		{
			printf("Not a valid command\n");
			delay(3000);
		}
		setcolor(WHITE);
		clrscr();
		cleardevice();
		anim=0;
		calTree(root,'\0',0,-1,-1,midx,30);
	}
}
//boundary animations
```

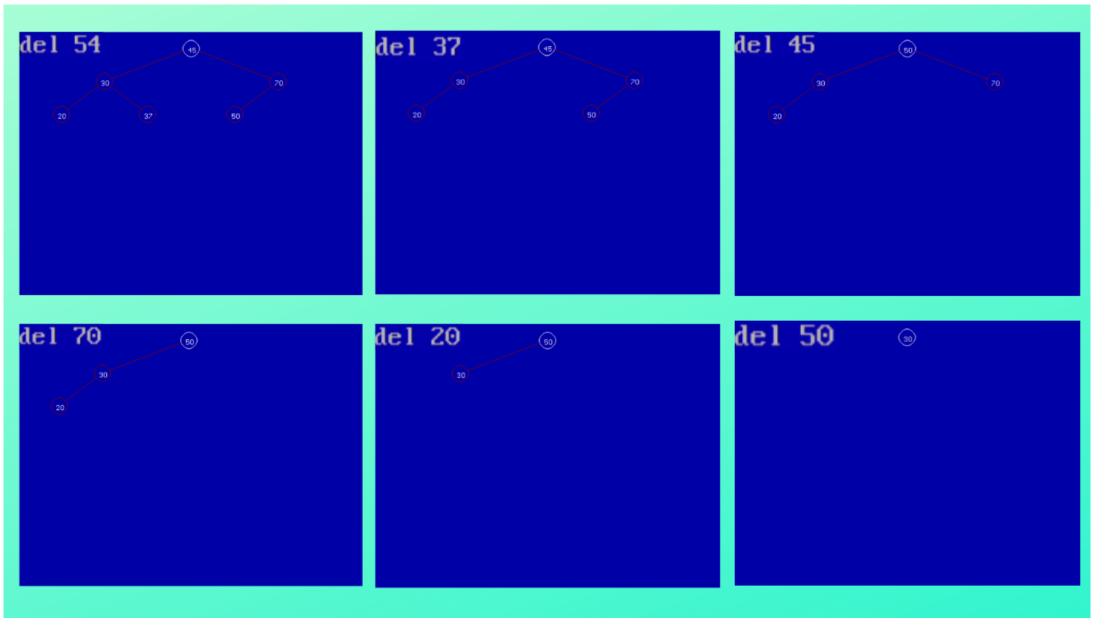
```
void boundary(int j,int k)
{
        int i,r=10;
        for(i=10;i<getmaxx();i++)
        {
                setcolor(j);
                circle(10,i,r);
                circle(getmaxx()-r,i,r);
                setcolor(k);
                circle(i,10,r);
                circle(i,getmaxy()-r,r);
                delay(1);
        }
}
```

Screenshots

Insert Operation:

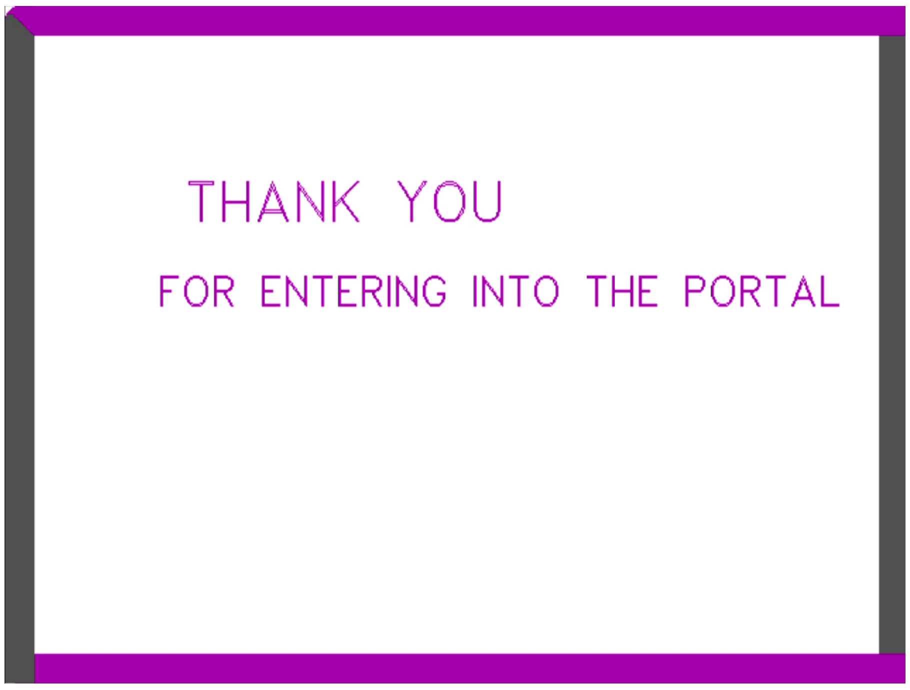


Delete Operation:



Working of the programme

- When you run the programme, first of all Project Title appears on the screen.
- After this user can add elements by the ins keyword. First(root) node is displayed by white colour and other nodes are displayed by red color
- Elements are arranged by BST properties.
- For Deleting node users have to use del keyword.
- To exit from the portal use exit keyword.
- After the completion of the animation, the screen is cleared and a message indicating `Thank You For Entering Into The Portal` Thank You For Entering Into The Portal appears on the screen.



```
THANK YOU
FOR ENTERING INTO THE PORTAL
```

Resources used

► graphics.h header file, winbgim.h header file and libbgi.a

► Turbo C++ 3.7.8.9

► Youtube:

1. Using graphics in C++:

   VCR Games (Video tutorials):
   https://youtube.com/playlist?list=PL5UFsTza4wWSNhe0xuO6ELw7ORU-UHNDO

2. Bresenham's Algorithm :

   Abdul Bari (Video Lecture):

   https://www.youtube.com/watch?v=RGB-wlatStc

* * * * * *