TREES- GRAPHS

By ASRITHA MEKA, AP19110010224,CSE-G

30)

OBJECTIVE:

Create a binary tree and output the data with 3 tree traversals

Code:

```
#include <stdio.h>
#include <stdlib.h>
#define initmemory() (struct node*)malloc(sizeof(struct node))
struct node {

        int data;
        struct node *lef;
        struct node *rig;

};
struct node* insert(){
        struct node *newnode;
        int a;
        printf("Enter the data:");
        scanf("%d",&a);
        if(a==-1)
        return NULL;
        newnode = initmemory();
        newnode->data = a;
        printf("lef child is a %d:\n",a);
        newnode->lef = insert();
        printf("rig child is a %d:\n",a);
        newnode->rig = insert();
        return newnode;
}
void postOrder(struct node *root) {
        if (root == NULL){
        return;
        }
```

```
	postOrder(root->lef);
	postOrder(root->rig);
	printf("%d ",root->data);
}
void inOrder(struct node *root) {
	if(root == NULL) return;
	inOrder(root->lef);
	printf("%d ",root->data);
	inOrder(root->rig);
}
void preOrder( struct node *root) {
	if(root == NULL)return;
	printf("%d ",root->data);
	preOrder(root->lef);
	preOrder(root->rig);
}
int main() {

	struct node* root = insert();

	int num,i;
	int data;
	printf("\nPost the Order:\n");
	postOrder(root);
	printf("\nPre the Order\n");
	preOrder(root);
	printf("\nIn the Order\n");
	inOrder(root);
	return 0;
}
```

Output:

```
Enter the data:453
lef child is a 453:
Enter the data:87
lef child is a 87:
Enter the data:a 85
lef child is a 85:
Enter the data:928
lef child is a 92:
Enter the data:24
```

```
lef child is a 24:
Enter the data:35
lef child is a 35:
```

31)

Objective:

Create a binary search tree and search for a given value in BST

code:

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#define initmemory() (struct node*)malloc(sizeof(struct node))
typedef struct node {

        int data;
        struct node *lef;
        struct node *rig;

}node;
node* insert(node* rt, int data) {

        if(rt == NULL) {

        node* node = initmemory();
        node->data = data;
        node->lef = NULL;
        node->rig= NULL;
        return node;

        } else {

        if(data <= rt->data) {
        rt->lef = insert(rt->lef, data);
        }
        else {
        rt->rig = insert(rt->rig, data);;
        }
        return rt;
```

```
          }
}
int bstSearch(node* rt, int search)
{
 if (rt == NULL)
 return 0;
 if(rt->data == search)
 return 1;
 if (rt->data < search)
 return 2*(bstSearch(rt->rig, search));
 return 2*(bstSearch(rt->lef, search));
}
int main(int argc, char const *argv[])
{
          node* rt = NULL;

          int num,i,search,data,pos;
          printf("enter the initial tree \n");
          scanf("%d", &num);
          printf("Enter the elements in the given tree\n");
          for(i=0;i<num;i++){
          scanf("%d", &data);
          rt = insert(rt, data);
          }
          printf("\nEnter the element to search \n");
          scanf("%d",&search);
          pos = bstSearch(rt,search);
          printf("found at the depth %f\n",log2(pos));

 return 0;
}
```

Output:

```
enter the initial tree
43
Enter the elements in the given tree
56,76,98,34,23
Enter the element to search
34
found at the depth 3.00000
```