Binary Search Tree

Node & Functions Used :-

```
struct Node
{
int data;
struct Node *left;
struct Node *right;
};
struct Node *root=NULL;
void insert(int);
void delete(int);
void inorder();
void preorder();
void postorder();
void search(int);
void findMin();
void findMax();
int getSuccessor(struct Node *);
```

Insertion():-

```
void insert(int data)
{
struct Node *temp,*parent,*current;
temp=(struct Node *)malloc(sizeof(struct Node));
temp->data=data;
temp->left=NULL;
temp->right=NULL;
if(root==NULL)
{
//First Node of tree
root=temp;
}
else
{
current=root;
while(current)
{
parent=current;
if(data>current->data)
{
//Go to right side
current=current->right;
}
else
{
```

```
//Go to left side
current=current->left;
}
}
if(data>parent->data)
{
//Go to right
parent->right=temp;
}
else
{
//Go to left
parent->left=temp;
}
}
}
```

Deleteion():-

```
void delete(int data)
{
if(root==NULL) return;
struct Node *curr,*parent,*rootNode;
curr=root;
parent=NULL;
bool done=0;
while(1)
{
if(data==root->data)
{
root=NULL;
free(root);
done=1;
break;
}
if(data>curr->data)
{
//Go to right
if(curr->right!=NULL)
{
parent=curr;
curr=curr->right;
}
else
{
printf("No node is present with this value\n");
break;
}
}
else if(data<curr->data)
{
//Go to left
if(curr->left!=NULL)
{
parent=curr;
```

```
curr=curr->left;
}
else
{
printf("No node is present with this value\n");
break;
}
}
else
{
if(curr->left==NULL && curr->right==NULL)
{
if(parent==NULL)
{
rootNode=NULL;
done=1;
break;
}
else if(parent->left->data==curr->data)
{
parent->left=NULL;
free(curr);
done=1;
break;
}
else
{
parent->right=NULL;
free(curr);
done=1;
break;
}
}
else if(curr->left==NULL)
{
if(parent==NULL)
{
rootNode=curr->right;
done=1;
break;
}
else if(parent->left->data==curr->data)
{
parent->left=curr->right;
free(curr);
done=1;
break;
}
else
{
parent->right=curr->right;
free(curr);
done=1;
```

```
break;
}
}
else if(curr->right==NULL)
{
if(parent==NULL)
{
rootNode=curr->left;
done=1;
break;
}
else if(parent->left->data==curr->data)
{
parent->left=curr->left;
free(curr);
done=1;
break;
}
else
{
parent->right=curr->left;
free(curr);
done=1;
break;
}
}
else
{
int successorNodeValue=getSuccessor(curr->right);
curr->data=successorNodeValue;
done=1;
break;
}
}
}//while(1) ends here..
if(done)
{
printf("%d deleted from tree\n",data);
}
else
{
printf("Element not deleted\n");
}
}//delete() function ends here..
```

InOrder():-

```
void inorder()
{
int size=1000;
struct Node *stack[size];
int top=size;
if(root==NULL) return;
struct Node *current=root;
```

```
bool done=0;
while(!done)
{
if(current!=NULL)
{
top--;
stack[top]=current;
current=current->left;
}
else
{
if(top!=size)
{
current=stack[top];
top++;
printf("%d ",current->data);
current=current->right;
}
else done=1;
}
}//while loop ending
}
```

preorder():-

```
void preorder()
{
if(root==NULL) return;
struct Node *current=root;
int size=1000,top;
top=size;
struct Node *stack[size];
bool done=0;
while(!done)
{
if(current!=NULL)
{
printf("%d ",current->data);
top--;
stack[top]=current;
current=current->left;
}
else
{
if(top!=size)
{
current=stack[top];
top++;
current=current->right;
}
else done=1;
}
}//while loop ending
}
```

postOrder():-

```
void postorder()
{
if(root==NULL) return;
int size=1000,top;
top=size;
struct Node *stack[size];
struct Node *current=root;
do
{
//Move to leftmost node
while(current)
{
//If current has right , push it and then push current into stack.
if(current->right)
{
top--;
stack[top]=current->right;
}
top--;
stack[top]=current;
current=current->left;
}
current=stack[top];
top++;
if(current->right && stack[top]==current->right)
{
stack[top]=current;
current=current->right;
}
else
{
printf("%d ",current->data);
current=NULL;
}
}while(top!=size);
}
```

Search():-

```
void search(int data)
{
if(root==NULL) return;
int size=1000;
struct Node *stack[size];
int top=size;
struct Node *current=root;
bool done=0;
bool found=0;
while(!done)
{
if(current!=NULL)
{
top--;
```

```
stack[top]=current;
current=current->left;
}
else
{
if(top!=size)
{
current=stack[top];
top++;
if(data==current->data)
{
found=1;
break;
}
current=current->right;
}
else done=1;
}
}//while loop ending
if(found)
{
printf("Element found\n");
}
else
{
printf("Element not found\n");
}
```

```
}
```

FindMin():-

```
void findMin()
{
if(root==NULL) return;
struct Node *temp;
temp=root;
while(temp)
{
temp=temp->left;
}
printf("%d is min\n",temp->data);
}
```

FindMax():-

```
void findMax()
{
if(root==NULL) return;
struct Node *temp;
temp=root;
while(temp)
{
temp=temp->right;
}
printf("%d is max\n",temp->data);
}
```

GetSuccessor():-

```
int getSuccessor(struct Node *curr)
{
while(1)
{
if(curr->left!=NULL)
{
curr=curr->left;
}
else
{
break;
}
}
return curr->data;
}
```