BINARY SEARCH TREE

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
struct node *left;
int element;
struct node *right;
};
typedef struct node Node;
Node *Insert(Node *Tree, int e);
void Find(Node *Tree, int e);
void FindMin(Node *Tree);
void FindMax(Node *Tree);
int main()
{
Node *Tree = NULL;
int n, i, e, ch;
printf("Enter number of nodes in the tree : ");
scanf("%d", &n);
printf("Enter the elements :\n");
for (i = 1; i <= n; i++)
{
scanf("%d", &e);
Tree = Insert(Tree, e);
}
do
{
printf("1. Find \n2. Find Min \n3. Find Max \n4. Exit\n");
printf("Enter your choice : ");
scanf("%d", &ch);
switch (ch)
{
case 1:
printf("Enter the element to find : ");
scanf("%d", &e);
Find(Tree, e);
printf("\n");
break;
case 2:
FindMin(Tree);
break;
case 3:
FindMax(Tree);
break;
}
} while (ch <= 3);
return 0;
}
Node *Insert(Node *Tree, int e)
{
Node *NewNode = malloc(sizeof(Node));
if (Tree == NULL)
{
NewNode->element = e;
NewNode->left = NULL;
NewNode->right = NULL;
Tree = NewNode;
}
```

```
else if (e < Tree->element)
Tree->left = Insert(Tree->left, e);
else if (e > Tree->element)
Tree->right = Insert(Tree->right, e);
return Tree;
}
void Find(Node *Tree, int e)
{
if (Tree == NULL)
printf("Element is not found...!");
else if (e < Tree->element)
Find(Tree->left, e);
else if (e > Tree->element)
Find(Tree->right, e);
else
printf("Element is found...!");
}
void FindMin(Node *Tree)
{
if (Tree == NULL)
printf("Tree is empty...!");
else if (Tree->left == NULL)
printf("%d\n", Tree->element);
else
FindMin(Tree->left);
}
void FindMax(Node *Tree)
{
if (Tree == NULL)
printf("Tree is empty...!");
else if (Tree->right == NULL)
printf("%d\n", Tree->element);
else
FindMax(Tree->right);
}
```

OUTPUT

```
Enter number of nodes in the tree : 6
Enter the elements :
6
2
8
1
4
3
1. Find
2. Find Min
3. Find Max
4. Exit
Enter your choice : 1
Enter the element to find : 2
Element is found...!
1. Find
2. Find Min
3. Find Max
4. Exit
Enter your choice : 1
Enter the element to find : 9
Element is not found...!
1. Find
2. Find Min
```

3. Find Max
4. Exit
Enter your choice : 2
1
1. Find
2. Find Min
3. Find Max
4. Exit
Enter your choice : 3
8
1. Find
2. Find Min
3. Find Max
4. Exit
Enter your choice : 4