Experiment B6

```
#include <iostream>
using namespace std;

struct node
{
    int key;
    struct node *left,*right;
};


struct node *insert(struct node *root,int value)
{
    if(root==NULL)
    {
        root=new node;
        root->key=value;
        root->left=NULL;
        root->right=NULL;
        return root;
    }

    else if (value == root->key)
    {
        return root;
    }
    else
    {
        if(root->key < value)
            root->right=insert(root->right,value);
        else
        {
            if(root->key > value)
                root->left=insert(root->left,value);
        }
    }
    return root;
}


void inorder(struct node *root)
{
    if(root != NULL)
    {
        inorder(root->left);
        cout<<"\t"<<root->key;
```

```
        inorder(root->right);
    }
    return;
}

void postorder(struct node *root)
{
    if(root != NULL)
    {
        postorder(root->left);
        postorder(root->right);
        cout<<"\t"<<root->key;
    }
    return;
}

void preorder(struct node *root)
{
    if(root != NULL)
    {
        cout<<"\t"<<root->key;
        preorder(root->left);
        preorder(root->right);
    }
    return;
}

struct node *search(struct node *root,int value)
{
    if (root == NULL || root->key == value)
       return root;

    if (root->key < value)
       return search(root->right, value);

    return search(root->left, value);
}
struct node *minimumval(struct node *root)
{
    struct node *current=root;
    while(current->left !=NULL)
        current=current->left;
    return current;
}

struct node *maximumval(struct node *root)
{
   struct node *current=root;
```

```
    while(current->right !=NULL)
        current=current->right;
    return current;
}

struct node *swapnodes(struct node *root)
{
    node *temp;
    if(root == NULL)
        return NULL;
    temp = root->left;
    root->left = root->right;
    root->right = temp;
    swapnodes(root->left);
    swapnodes(root->right);
    return root;

}

int longestpath( node *root)
{
if (root==NULL){
return 0;
}
int leftlong=longestpath(root->left);
int rightlong=longestpath(root->right);
return max(leftlong,rightlong)+1;
}


int main()
{
        int choice=0,value=0,value1=0,d;
        struct node *root=NULL,*searchh=NULL,*position=NULL;
    do
    {

        cout<<"\n\n";
        cout<<"\n ------ Binary Search Tree ------";
        cout<<"\n [1] Insertion ";
        cout<<"\n [2] Search ";
        cout<<"\n [3] Traversals ";
        cout<<"\n [4] Minimum Value ";
        cout<<"\n [5] Maximum Value";
        cout<<"\n [6] Number of nodes in longest path:";
        cout<<"\n [7] Swap nodes:";
        cout<<"\n [0] Exit ";
```

```
cout<<"\n Enter the choice: ";
cin>>choice;

switch(choice)
{
   case 1:
         cout<<"\n Insertion..!";
         cout<<"\n Enter the element to be inserted: ";
         cin>>value;
         root=insert(root,value);

         break;

   case 2:
         cout<<"\n Search..!";
         cout<<"\n Enter the element to be searched: ";
         cin>>value;
         searchh=search(root,value);
         if(searchh == NULL)
            cout<<"\n Key not found!";
         else
         {
            cout<<"\n"<<" Key "<<searchh->key<<" Found!";
         }
         break;
   case 3:
         cout<<"\n Traversals..!";
         cout<<"\n Inorder: ";
         inorder(root);
         cout<<"\n Preorder: ";
         preorder(root);
         cout<<"\n Postorder: ";
         postorder(root);
         break;
  case 4:
         cout<<"\n Minimum Value..!";
         if(root == NULL)
            cout<<"\n No minimum values in empty tree";
         else
         {
            value=minimumval(root)->key;
            cout<<"\n Smallest value in the tree: "<<value;
        }
         break;
   case 5:
         cout<<"\n Maximum Value..!";
         if(root == NULL)
            cout<<"\n No maximum values in empty tree";
```

```
                else
                {
                    value=maximumval(root)->key;
                    cout<<"\n Largest value in the tree: "<<value;
                }

                break;
            case 6:
                d=longestpath(root);
                cout<<"The no. of nodes in the longest path are:"<<d<<endl;
                break;
            case 7:
               swapnodes(root);
               break;
            case 0:
                cout<<"\n Exiting..!";
                break;
            default:
                cout<<"\n Invalid Choice!";
                break;
        }

    }while(choice!=0);

    return 0;
}
```