Assignment #1

Subject:	*Data Structure & Algo.*
Topic:	Array+Link BST
Submitted to:	Sir Zeeshan
Submitted by:	Muhammad Shahzaib
Roll No.	109
Date:	18-11-2021

ArrayBST.h:

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

struct Node
{
   int data;
   Node *left, *right;
   Node(int val)
   {
   data = val;
   Node *left = NULL;
   Node *right = NULL;
   }
};

namespace Arr{
void postOrderArr(Node *root)
{
   if (root != NULL)
   {
      postOrderArr(root->left);
      postOrderArr(root->right);
      cout << root->data << " ";
   }
}

void sortArr(int size)
{
   int arr[size], i, j, temp;
   for (i = 0; i < size; i++)
   {
      cout << "Enter the Number : ";
      cin >> arr[i];
   }

   for (i = 0; i < size; ++i)
   {
      for (j = i + 1; j < size; ++j)
      {
          if (arr[i] > arr[j])
          {
              temp = arr[i];
              arr[i] = arr[j];
              arr[j] = temp;
          }
      }
   }
}

Node *ArrToBST(int arr[], int start, int end)
{

   if (start>end)
   {
      return NULL;
   }
   int mid = (start + end) / 2;

   Node *root = new Node(arr[mid]);

   root->left = ArrToBST(arr, start, mid - 1);

   root->right = ArrToBST(arr, mid + 1, end);

   return root;


}

void preOrderArr(Node* root)

{

   if (root==NULL)

   {

      return;

   }


   cout<<root->data<<" ";

   preOrderArr(root->left);

   preOrderArr(root->right);

}

Node *searchBSTArr(Node *root, int key)

{

   if (root == NULL)

   {

      return NULL;

   }

   if (key == root->data)


      return root;


   else if (key < root->data)
```

```
        return searchBSTArr(root->left, key);


    else
        return searchBSTArr(root->right, key);
}
Node *searchInBSTArr(int key, Node *root)
{


    Node *check = searchBSTArr(root, key);
    if (check != NULL)
    {
        cout << "\nElement " << check->data << " found in this BST..." << endl;
    }
    else
        cout << "\nElement Not Found..." << endl;
}
Node *inOrderPredecessor(Node *root) //right Most child of left subtree
{
    root = root->left;
    while (root->right != NULL)
    {
        root = root->right;
    }
    return root;
}
Node *deleteNode(Node *root, int value)
{
    Node *iPre;
    if (root == NULL)
    {
        return NULL;
    }

    if (root->left == NULL && root->right == NULL)
    {
        delete root;
        return NULL;
    }

    if (value < root->data)
    {
        root->left = deleteNode(root->left, value);
    }
    else if (value > root->data)
    {
        root->right = deleteNode(root->right, value);
    }
    else
    {
        iPre = inOrderPredecessor(root);
        root->data = iPre->data;
        root->left = deleteNode(root->left, iPre->data);
    }
    return root;
}
void inOrderArr(Node *root)
{
    if (root == NULL)

    {

        return;

    }

        inOrderArr(root->left);

        cout << root->data << " ";

        inOrderArr(root->right);


}

}
```

LinkedBST.h:

```
//BST-ASSIGNMENT_RollNo_109_Shahzaib
#include <iostream>
using namespace std;

struct node
{
   int data;
   node *left;
   node *right;
};
namespace link{
node *createNewNode(int value)
{
   struct node *newNode = new node;
   newNode->data = value;
   newNode->left = NULL;
   newNode->right = NULL;
   return newNode;
}

void preOrder(node *root)
{
   if (root != NULL)
   {
      cout << root->data << " ";
      preOrder(root->left);
      preOrder(root->right);
   }
```

```
}
void postOrder(node *root)
{
   if (root != NULL)
   {
      preOrder(root->left);
      preOrder(root->right);
      cout << root->data << " ";
   }
}
void inOrder(node *root)
{
   if (root != NULL)
   {
      inOrder(root->left);
      cout << root->data << " ";
      inOrder(root->right);
   }
}

node *searchBST(node *root, int key)
{
   if (root == NULL)
   {
      return NULL;
   }
   if (key == root->data)

      return root;
```

```
    else if (key < root->data)

        return searchBST(root->left, key);

    else
        return searchBST(root->right, key);
}

node *searchInBST(int key, node *root)
{
    node *check = searchBST(root, key);
    if (check != NULL)
    {
        cout << "\nElement " << check->data << " found in this BST..." << endl;
    }
    else
        cout << "\nElement Not Found..." << endl;
}

void insertInBST(node *root, int key)
{
    node *prevNode = NULL;

    while (root != NULL)
    {
        prevNode = root;
        if (key == root->data)
        {
            cout << "\nCan't Insert, the given Element '" << key << "' already exists in this BST!"
<< endl;
            return;
        }
        else if (key < root->data)
        {
            root = root->left;
        }
        else
            root = root->right;
    }
    node *newNode = createNewNode(key);
    if (key < prevNode->data)
    {
        prevNode->left = newNode;
    }
    else
        prevNode->right = newNode;

    cout << "\nElement Added in BST successfuly...! " << endl;
}

node *inOrderPredecessor(node *root) //right Most child of left subtree
{
    root = root->left;
    while (root->right != NULL)
    {
        root = root->right;
    }
    return root;
```

```
}
node *deleteNode(node *root, int value)
{
    node *iPre;
    if (root == NULL)
    {
        return NULL;
    }

    if (root->left == NULL && root->right == NULL)
    {
        delete root;
        return NULL;
    }

    if (value < root->data)
    {
        root->left = deleteNode(root->left, value);
    }
    else if (value > root->data)
    {
        root->right = deleteNode(root->right, value);
    }
    else
    {
        iPre = inOrderPredecessor(root);
        root->data = iPre->data;
        root->left = deleteNode(root->left, iPre->data);
    }
```

```
   return root;
}
}
```

main.cpp

```
//BST-ASSIGNMENT_RollNo_109_Shahzaib

#include <iostream>

#include "LinkedBST.h"

#include "ArrayBST.h"

using namespace Arr;

using namespace link;

using namespace std;


Node *sortedArrayToBST(int size)

{


    int *arr = new int[size];


    int i, j, temp;

    for (i = 0; i < size; i++)

    {

        cout << "Enter the Number : ";

        cin >> arr[i];

    }


    for (i = 0; i < size; ++i)

    {

        for (j = i + 1; j < size; ++j)

        {

            if (arr[i] > arr[j])

            {

                temp = arr[i];

                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
    }

    Node *root = Arr::ArrToBST(arr, 0, size - 1);
    return root;
}

int main()
{
    while(true)
    {
        int proceed;
        cout << "please select 1 or 2 to proceed\n1.Link BST\n2.Array BST\n3.Exit" << endl;
        cin >> proceed;

        if (proceed == 1)
        {
            node *p = link::createNewNode(9);
            node *p1 = link::createNewNode(3);
            node *p2 = link::createNewNode(11);
            p->left = p1;
            p->right = p2;

            while (true)
            {
                cout << "\nFollowing Basic Operations are availabe in Link BST Data Structure:" <<
endl;
                cout << "1. Insert Element in BST" << endl;
                cout << "2. Find Element in BST" << endl;

                cout << "3. Remove Element in BST" << endl;

                cout << "4. Print BST" << endl;

                cout << "5. Exit from Main Menu" << endl;


                int enter;

                cin >> enter;

                switch (enter)

                {

                case 1:


                   while (true)


                   {

                      int value;

                      char check;

                      cout << "Please enter element to insert in BST" << endl;

                      cin >> value;

                      link::insertInBST(p, value);

                      cout << "Continue [Y/Any other key to exit]: ";

                      cin >> check;

                      if (check == 'y' || check == 'Y')

                      {

                         continue;

                      }

                      else

                         break;

                   }

                   break;
```

```
case 2:
    int key;
    cout << "Enter key YOU Want to search: ";
    cin >> key;
    link::searchInBST(key, p);
    break;
case 3:
    int del;
    cout << "Enter an element YOU Want to Delete: ";
    cin >> del;
    node *test;
    test = link::searchBST(p, del);
    if (test != NULL)
    {
        link::deleteNode(p, del);
        cout << "Element deleted Sucessfully..." << endl;
    }
    else
        cout << "This Element doesn't Exists in BST..." << endl;

    break;
case 4:
    int print;
    cout << "In Which Order Do you Want to Print:" << endl;
    cout << "1. Pre-Order" << endl;
    cout << "2. Post-Order" << endl;
    cout << "3. In-Order" << endl;
    cout << "4. Abort the Printing" << endl;
    cin >> print;
    if (print == 1)
    {
        cout << "Print Pre-Order: ";
        link::preOrder(p);
        cout << endl;
    }
    else if (print == 2)
    {
        cout << "Print Post-Order: ";
        link::postOrder(p);
        cout << endl;
    }
    else if (print == 3)
    {
        cout << "Print In-Order: ";
        link::inOrder(p);
        cout << endl;
    }
    else if (print == 4)
        break;
    else
        continue;
    break;
case 5:
    cout << "\nExiting Program..." << endl;
    return false;
    break;
default:
```

```
                cout << "Choose b/w 1 to 5..." << endl;
                break;
            }
        }
    }
    else if (proceed == 2)
    {

        int size;
        cout << "Enter the Size of Array: ";
        cin >> size;
        Node *root = sortedArrayToBST(size);

        while (true)
        {
            cout << "\nFollowing Basic Operations are availabe in Array BST Data Structure:" <<
endl;
            cout << "1. Insert Element in BST" << endl;
            cout << "2. Find Element in BST" << endl;
            cout << "3. Remove Element in BST" << endl;
            cout << "4. Print BST" << endl;
            cout << "5. Exit from Main Menu" << endl;

            int enter;
            cin >> enter;
            switch (enter)
            {
            case 1:

                break;
```

```
case 2:
   int key;
   cout << "Enter key YOU Want to search: ";
   cin >> key;
   Arr::searchInBSTArr(key, root);
   break;
case 3:

   break;
case 4:
   int print;
   cout << "In Which Order Do you Want to Print:" << endl;
   cout << "1. Pre-Order" << endl;
   cout << "2. Post-Order" << endl;
   cout << "3. In-Order" << endl;
   cout << "4. Abort the Printing" << endl;

   cin >> print;
   if (print == 1)
   {
      cout << "Print Pre-Order: ";
      Arr::preOrderArr(root);
      cout << endl;
   }
   else if (print == 2)
   {
      cout << "Print Post-Order: ";
      Arr::postOrderArr(root);
      cout << endl;
```

```
            }
            else if (print == 3)
            {
                cout << "Print In-Order: ";
                Arr::inOrderArr(root);
                cout << endl;
            }
            else if (print == 4)
                break;
            else
                continue;
            break;
        case 5:
            cout << "\nExiting Program..." << endl;
            return false;
            break;
        default:
            cout << "Choose b/w 1 to 5..." << endl;
            break;
        }
    }
}
else if(proceed==3)
{
    cout << "\nExiting Program..." << endl;
    return false;
}
elsey
continue;
```

```
}
	return 0;
}
```

ArrayBST.cpp

```
#include <iostream>
#include"ArrBST.h"
using namespace Arr;
using namespace std;

Node* sortedArrayToBST(int size){

   int* arr = new  int[size];

   int i, j, temp;
   for (i = 0; i < size; i++)
   {
      cout << "Enter the Number : ";
      cin >> arr[i];
   }

   for (i = 0; i < size; ++i)
   {
      for (j = i + 1; j < size; ++j)
      {
         if (arr[i] > arr[j])
         {
            temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
         }
      }
```

```
    }


    Node* root =Arr::ArrToBST(arr,0,size-1);
    return root;
}


int main()
{
    int size;
    cout<<"Enter the Size of Array: ";
    cin>>size;
    Node* root = sortedArrayToBST(size);


    while (true)
    {
        cout << "\nFollowing Basic Operations are availabe in Array BST Data Structure:" <<
endl;
        cout << "1. Insert Element in BST" << endl;
        cout << "2. Find Element in BST" << endl;
        cout << "3. Remove Element in BST" << endl;
        cout << "4. Print BST" << endl;
        cout << "5. Exit from Main Menu" << endl;


        int enter;
        cin >> enter;
        switch (enter)
        {
        case 1:


            break;
```

```
case 2:
   int key;
   cout << "Enter key YOU Want to search: ";
   cin >> key;
   Arr::searchInBSTArr(key, root);
   break;
case 3:


   break;
case 4:
   int print;
   cout << "In Which Order Do you Want to Print:" << endl;
   cout << "1. Pre-Order" << endl;
   cout << "2. Post-Order" << endl;
   cout << "3. In-Order" << endl;
   cout << "4. Abort the Printing" << endl;


   cin >> print;
   if (print == 1)
   {
      cout << "Print Pre-Order: ";
      Arr::preOrderArr(root);
      cout << endl;
   }
   else if (print == 2)
   {
      cout << "Print Post-Order: ";
      Arr::postOrderArr(root);
      cout << endl;
```

```
            }
            else if (print == 3)
            {
                cout << "Print In-Order: ";
                Arr::inOrderArr(root);
                cout << endl;
            }
            else if (print == 4)
                break;
            else
                continue;
            break;
        case 5:
            cout << "\nExiting Program..." << endl;
            return false;
            break;
        default:
            cout<<"Choose b/w 1 to 5..."<<endl;
            break;
        }
    }

    return 0;
}
```

LinkedBST.cpp

```
#include <iostream>

#include "LinkedBST.h"

using namespace link;

using namespace std;


int main()

{

   node *p = link::createNewNode(9);

   node *p1 = link::createNewNode(3);

   node *p2 = link::createNewNode(11);

   p->left = p1;

   p->right = p2;


   while (true)

   {

      cout << "\nFollowing Basic Operations are availabe in BST Data Structure:" << endl;

      cout << "1. Insert Element in BST" << endl;

      cout << "2. Find Element in BST" << endl;

      cout << "3. Remove Element in BST" << endl;

      cout << "4. Print BST" << endl;

      cout << "5. Exit from Main Menu" << endl;


      int enter;

      cin >> enter;

      switch (enter)

      {

      case 1:
```

```
    while (true)

    {
        int value;
        char check;
        cout << "Please enter element to insert in BST" << endl;
        cin >> value;
        link::insertInBST(p, value);
        cout << "Continue [Y/Any other key to exit]: ";
        cin >> check;
        if (check == 'y' || check == 'Y')
        {
            continue;
        }
        else
            break;
    }
    break;
case 2:
    int key;
    cout << "Enter key YOU Want to search: ";
    cin >> key;
    link::searchInBST(key, p);
    break;
case 3:
    int del;
    cout << "Enter an element YOU Want to Delete: ";
    cin >> del;
    node *test;
    test = link::searchBST(p,del);
    if (test!=NULL)
    {
        link::deleteNode(p, del);
        cout << "Element deleted Sucessfully..." << endl;
    }
    else
        cout << "This Element doesn't Exists in BST..." << endl;

    break;
case 4:
    int print;
    cout << "In Which Order Do you Want to Print:" << endl;
    cout << "1. Pre-Order" << endl;
    cout << "2. Post-Order" << endl;
    cout << "3. In-Order" << endl;
    cout << "4. Abort the Printing" << endl;

    cin >> print;
    if (print == 1)
    {
        cout << "Print Pre-Order: ";
        link::preOrder(p);
        cout << endl;
    }
    else if (print == 2)
    {
        cout << "Print Post-Order: ";
        link::postOrder(p);
```

```
                cout << endl;
            }
            else if (print == 3)
            {
                cout << "Print In-Order: ";
                link::inOrder(p);
                cout << endl;
            }
            else if (print == 4)
                break;
            else
                continue;
            break;
        case 5:
            cout << "\nExiting Program..." << endl;
            return false;
            break;
        default:
            cout<<"Choose b/w 1 to 5..."<<endl;
            break;
        }
    }
    return 0;
}
```