Kathmandu University
Department of Computer Science and Engineering
Dhulikhel, Kavre

COMP 202

Lab Report 3

Submitted by
Ashish kumar khatri
Roll No: 28
CE 2nd Year/1st Sem

Submitted To,

Rajani Chulyadyo

Department of Computer Science and Engineering

- **AbstractBST.h**

```
class AbstractBST{

    public:
        virtual bool isEmpty() =0;
        virtual void add(int key,int value) = 0;
        virtual void max(int &output) = 0;
        virtual void min(int &output) = 0;
        virtual bool exists(int targetKey) = 0;
        virtual void inorder() = 0;
};
```

- **ArrayBST.h**

```
#pragma once
#include"AbstractBST.h"
#define MAX_NUM_NODES 120

class Node{
    public:
        int key;
        int value;
        Node();
        Node(int,int);
};


class ArrayBST:public AbstractBST{
    private:
        Node* array[MAX_NUM_NODES];
    public:
        ArrayBST();
        bool isEmpty() override;
        void add(int , int) override;
        void inorder() override;
        void inorder(int);
        void postOrder(int);
        void postOrder();
        void preOrder(int);
        void preOrder();
        void max(int&) override;
        void min(int&) override;
        bool exists(int) override;
};
```

- **ArrayBST.cpp**

```
#include"ArrayBST.h"
#include<iostream>

Node::Node(int key , int value)
   :key(key),value(value){}

ArrayBST::ArrayBST(){
   for(int i = 0 ; i<MAX_NUM_NODES ; i++)
      array[i] = NULL;
}

bool ArrayBST::isEmpty(){
   return array[1] == NULL;
}

void ArrayBST::add(int key , int value){
   Node* newNode = new Node(key,value);

   for(int i = 1 ; i < MAX_NUM_NODES ; ){
      if(array[i] == NULL){
         array[i] = newNode;
         std::cout << "inserted  key = " << array[i] -> key << " and value = " <<
array[i]->value<< std::endl;
         break;
      }else if(array[i]->key > key){
         i *= 2;
      }else if(array[i] -> key < key){
         i = i*2 + 1;
      }
   }
}

void ArrayBST::inorder(int i){
   if(array[i] == NULL) return;
   inorder(i*2);
   std::cout << "Key = " << array[i] -> key  << " and value = " << array[i] -> value <<
"\n";
   inorder(i*2+1);
```

```
}

void ArrayBST::postOrder(int i){
    if(array[i] == NULL) return;
    postOrder(i*2);
    postOrder(i*2+1);
    std::cout << "Key = " << array[i] -> key  << " and value = " << array[i] -> value <<
"\n";


}

void ArrayBST::preOrder(int i){
    if(array[i] == NULL) return;
    std::cout << "Key = " << array[i] -> key  << " and value = " << array[i] -> value <<
"\n";
    preOrder(i*2);
    preOrder(i*2+1);
}



void ArrayBST::inorder(){
    std::cout << "\nInorder of the BST is : ";
    if(array[1] == NULL){
        std::cout << " Empty BST" << std::endl;
    }else{
        std::cout << "\n";
        this -> inorder(1);
        std::cout << "\n";
    }
}

void ArrayBST::postOrder(){
    std::cout << "\nPostOrder of the BST is : ";
    if(array[1] == NULL){
        std::cout << " Empty BST" << std::endl;
    }else{
        std::cout << "\n";
```

```
        this -> postOrder(1);
        std::cout << "\n";
    }
}

void ArrayBST::preOrder(){
    std::cout << "\npreOrder of the BST is : ";
    if(array[1] == NULL){
        std::cout << " Empty BST" << std::endl;
    }else{
        std::cout << "\n";
        this -> preOrder(1);
        std::cout << "\n";
    }
}


void ArrayBST::max(int& output){
    if(this -> isEmpty()){
        std::cout << "Empty tree cannot find max value" << std::endl;
        return;
    }
    int i = 1;
    while( i < MAX_NUM_NODES ){
        if(array[i] == NULL){
            i = (i-1)/2;
            break;
        }
        i = 2*i + 1;
    }
    output = array[i] -> key;
}

void ArrayBST::min(int& output){
    if(this -> isEmpty()){
        std::cout << "Empty tree cannot find max value" << std::endl;
        return;
    }
    int i = 1;
    while( i < MAX_NUM_NODES ){
```

```
        if(array[i] == NULL){
            i = (i)/2;
            break;
        }
        i = 2*i;
    }
    output = array[i] -> key;
}


bool ArrayBST::exists(int target){
    if(this -> isEmpty()){
        std::cout << "Empty tree , target doesnot exists" << std::endl;
        return 0;
    }
    int i = 1;
    while(array[i] != NULL){
        if(array[i] -> key == target){
            return true;
        }else if(target > array[i] -> key) i = i*2 +1;
         else if(target < array[i] -> key) i = i*2;
    }
    return false;
}
```

- **main.cpp**

```
#include<iostream>
#include"ArrayBST.h"

int main(){
    ArrayBST* A = new ArrayBST();
    A -> inorder();
    std::cout << "\n";
    A -> add(10 , 10000);
    A->add(5 , 5000);
    A->add(2,2000);
    A->add(6,6000);
    A -> add(15 , 15000);
    A -> add(12 , 12000);
    A -> add(11 , 11000);
    A -> add(14 , 14000);
    A -> add(13 , 13000);
    A->inorder();
    A->postOrder();
    A->preOrder();
    int maxKey;
    A->max(maxKey);
    std::cout << "MaxKey value = " << maxKey<<std::endl;
    int minKey;
    A->min(minKey);
    std::cout << "MinKey value = " << minKey<<std::endl;
    std::cout << A->exists(5) << std::endl;
    std::cout << A->exists(30) << std::endl;

    return 0;
}
```