Name: **Kashif Ali**
Roll No: **20P-0648**
Section: **3D**
Lab 8 tasks

task1.cpp

```cpp
// Task 1 to check whether it is bst or not
#include<iostream>
using namespace std;

class Node{
    public:
        Node *left;
        Node *right;
        int data;
        Node(int val){
            data=val;
            this->left=NULL;
            this->right=NULL;
        }

};
Node *Insert(Node *root,int data){
    if(root==NULL){
        return new Node(data);
    }
    else if(root->data>data){
        root->left=Insert(root->left,data);

    } else{
        root->right=Insert(root->right,data);
    }
    return root;
}



int check(Node *root){

    if(root->left!=NULL && root->left->data<root->data){
        return 0;
    }

    if(root->right!=NULL && root->right->data>root->data){
        return 0;
    }
    if(!check(root->left)|| !check(root->right)){
        return 0;
    }
    return 1;
}
int main(){
    Node *root;
    root=NULL;
    root=Insert(root,21);
    Insert(root,12);
    Insert(root,13);

    cout<<endl;
    if(check(root)==NULL){
        cout<<" Marks= 10 ";
    }
    else{
        cout<<" Marks=0 ";
```

Task 1

code

output

```
kashiii@kashiii:~/Downloads/dslab8$ ./task1.exe

 Marks= 10
kashiii@kashiii:~/Downloads/dslab8$
```

task2a.cpp > BinaryTree > addnode(int, string)

```cpp
#include<iostream>
using namespace std;
class node{
    public:
        node *left;
        node *right;
        string data;
        node(string val){
            this->left=NULL;
            this->right=NULL;
            val=data;
        }
};
class BinaryTree{
    private:
        node *root;
    public:
        void addnode(int key,string name){
            node *tmp;
            tmp=new node(name);
            if(root==NULL){
                return new node(name);
            }
            else if(root->data>name){
                root->left=addnode(key, name);

            }
        }

};

```

Task 2 A code

Task #2.cpp

Task #2.cpp > main()

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

class node{
    public:
        node *left;
        node *right;
        int data;
        node(int val){
            data=val;
            this->left=NULL;
            this->right=NULL;
        }

};



node *Insert(node *root,int data){
    if(root==NULL){
        return new node(data);
    }
    else if(root->data>data){
        root->left=Insert(root->left,data);

    } else{
        root->right=Insert(root->right,data);
    }
    return root;
}

int main(){

    node *root;
    root=NULL;
root=Insert(root,15);
    Insert(root,10);
    Insert(root,8);
    Insert(root,20);
    Insert(root,25);
    Insert(root,12);
    Insert(root,16);

    // preorder(root);

}

```

Task 2 A

task2b.cpp > preorder(node *)

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

class node{
    public:
        node *left;
        node *right;
        int data;
        node(int val){
            data=val;
            this->left=NULL;
            this->right=NULL;
        }

};



node *Insert(node *root,int data){
    if(root==NULL){
        return new node(data);
    }
    else if(root->data>data){
        root->left=Insert(root->left,data);

    } else{
        root->right=Insert(root->right,data);
    }
    return root;
}
void preorder(node *root){
    if(root==NULL){
        return;
    }
    else{
        cout<<root->data<<" ";
        preorder(root->left);
        preorder(root->right);
    }
}

int main(){

    node *root;
    root=NULL;
root=Insert(root,15);
    Insert(root,10);
    Insert(root,8);
    Insert(root,20);
    Insert(root,25);
    Insert(root,12);
    Insert(root,16);

    preorder(root);
```

Task 2 b code

output

```
kashiii@kashiii:~/Downloads/dslab8$ ./task2b.exe
15 10 8 12 20 16 25 kashiii@kashiii:~/Downloads/dslab8$
```

task3a.cpp

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

class node{
    public:
        node *left;
        node *right;
        int data;
        node(int val){
            data=val;
            this->left=NULL;
            this->right=NULL;
        }

};
node *Insert(node *root,int data){
    if(root==NULL){
        return new node(data);
    }
    else if(root->data>data){
        root->left=Insert(root->left,data);

    } else{
        root->right=Insert(root->right,data);
    }
    return root;
}
void Inorder(node *root){
    if(root==NULL){
        return;
    }
    else{
        Inorder(root->left);
        cout<<root->data<<" ";
        Inorder(root->right);
    }
}
void pre_order(node *root){
    if(root==NULL){
        return;
    }
    else{
        cout<<root->data<<" ";
        pre_order(root->left);
        pre_order(root->right);
    }

}


int main(){
    node *root;
    root=NULL;
    root=Insert(root,21);
    Insert(root,12);
    Insert(root,13);
    cout<<"Inorder "<<endl;
    Inorder(root);
    cout<<endl;
    //check(root);

        cout<<endl;
        cout<<"pre order "<<endl;
        pre_order(root);
        }
```

Task 3 A code

output

```
kashiii@kashiii:~/Downloads/dslab8$ ./task3a.exe
Inorder
12 13 21

pre order
kashiii@kashiii:~/Downloads/dslab8$
```

task3b.cpp

```cpp
// C++ program to implement iterative preorder traversal
#include <bits/stdc++.h>

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


void iterativePreorder(node* root)
{
    if (root == NULL)
        return;


    stack<node*> nodeStack;
    nodeStack.push(root);


    while (nodeStack.empty() == false) {
        struct node* node = nodeStack.top();
        printf("%d ", node->data);
        nodeStack.pop();

        if (node->right)
            nodeStack.push(node->right);
        if (node->left)
            nodeStack.push(node->left);
    }
}

int main()
{

    struct node* root = newNode(10);
    root->left = newNode(8);
    root->right = newNode(2);
    root->left->left = newNode(3);
    root->left->right = newNode(5);
    root->right->left = newNode(2);
    iterativePreorder(root);
    return 0;
}
```

Task 3 B code

output

```
kashiii@kashiii:~/Downloads/dslab8$ g++ task3b.cpp -o task3
task3a.cpp  task3a.exe  task3b.cpp
kashiii@kashiii:~/Downloads/dslab8$ g++ task3b.cpp -o task3b.exe
kashiii@kashiii:~/Downloads/dslab8$ ./task3b.exe
10 8 3 5 2 2 kashiii@kashiii:~/Downloads/dslab8$
```

task4.cpp

task4.cpp > removeOutsideRange(node *, int, int)

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

class node
{   public:
    int key;
    node *left;
     node *right;
};

node* removeOutsideRange(node *root, int min, int max)
{
if (root == NULL)
    return NULL;
root->left = removeOutsideRange(root->left, min, max);
root->right = removeOutsideRange(root->right, min, max);


if (root->key < min)
{
    node *rChild = root->right;

    return rChild;
}
if (root->key > max)
{
    node *lChild = root->left;
    return lChild;
}
return root;
}
node* newNode(int num)
{
    node* temp = new node;
    temp->key = num;
    temp->left = temp->right = NULL;
    return temp;
}
```

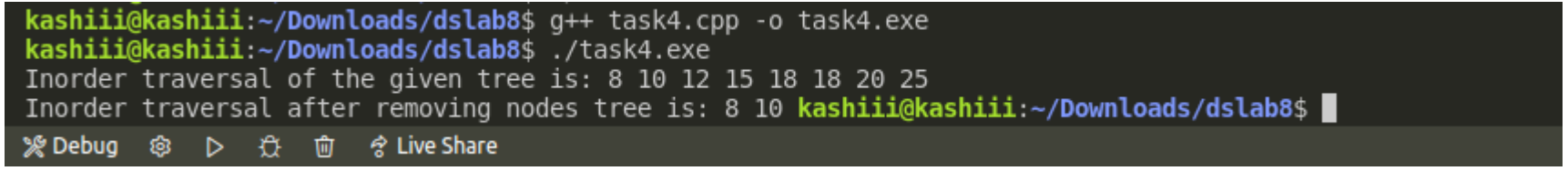
Task 4 code part a

task4.cpp > main()

```cpp
node* insert(node* root, int key)
{
    if (root == NULL)
    return newNode(key);
    if (root->key > key)
    root->left = insert(root->left, key);
    else
    root->right = insert(root->right, key);
    return root;
}

void inorderTraversal(node* root)
{
    if (root)
    {
        inorderTraversal( root->left );
        cout << root->key << " ";
        inorderTraversal( root->right );
    }
}
int main()
{
    node* root = NULL;
    root = insert(root, 15);
    root = insert(root, 10);
    root = insert(root, 20);
    root = insert(root, 8);
    root = insert(root, 12);
    root = insert(root, 18);
    root = insert(root, 25);
        root = insert(root, 18);


    cout << "Inorder traversal of the given tree is: ";
    inorderTraversal(root);

root = removeOutsideRange(root, 8, 10);

    cout << "\nInorder traversal after removing nodes tree is: ";
    inorderTraversal(root);

    return 0;
}
```

Task 4 code part b

Task 4 output



Thank you

...