question 1

67

32 78

18 56

13 20

9

in : 9 13 18 20 32 56 67 78

pre : 67 32 18 13 9 20 56 78

post: 9 13 20 18 56 32 78 67

Question 2

#include <iostream>

using namespace std;

template <typename t>

class queue

{

public:

t data;

queue \*next;

queue(t d)

{

this->next = NULL;

this->data = d;

}

void push(queue \*&front, queue \*&rear, t d)

{

queue \*n = new queue(d);

if (front == NULL)

{

front = n;

rear = n;

return;

}

else

{

rear->next = n;

rear = rear->next;

}

}

void pop(queue \*&front, queue \*&rear)

{

if (front == rear)

{

front = rear = NULL;

}

else

{

queue \*to\_del = front;

front = front->next;

delete (to\_del);

}

}

bool isempty(queue \*front)

{

return front == NULL ? true : false;

}

t front(queue \*&front)

{

if (front != NULL)

{

return front->data;

}

}

};

// bst

class bst

{

public:

int data;

bst \*left;

bst \*right;

bst(int d)

{

this->data = d;

this->left = NULL;

this->right = NULL;

}

bst \*insert\_into\_bst(bst \*root, int d)

{

if (root == NULL)

{

bst \*n = new bst(d);

return n;

}

if (root->data < d)

{

root->right = insert\_into\_bst(root->right, d);

}

else

{

root->left = insert\_into\_bst(root->left, d);

}

return root;

}

void create(bst \*&root)

{

cout << " enter the data : (0 to exit ) : ";

int d;

cin >> d;

while (d)

{

root = root->insert\_into\_bst(root, d);

cout << " enter the data : (0 to exit ) : ";

cin >> d;

}

}

void inorder(bst \*root)

{

if (root == NULL)

return;

inorder(root->left);

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

inorder(root->right);

}

void level\_order(bst \*root)

{

queue<bst \*> \*qf = NULL;

queue<bst \*> \*qr = NULL;

qf->push(qf, qr, root);

qf->push(qf, qr, NULL);

while (!qf->isempty(qf))

{

bst \*front = qf->front(qf);

qf->pop(qf, qr);

if (front == NULL)

{

if (!qf->isempty(qf))

{

qf->push(qf, qr, NULL);

}

cout << endl;

}

else

{

cout<<front->data<<" ";

if (front->left)

{

qf->push(qf, qr, front->left);

}

if (front->right)

{

qf->push(qf, qr, front->right);

}

}

}

}

};

int main()

{

bst \*root = NULL;

root->create(root);

cout << "\ninorder : ";

root->inorder(root);

cout<<endl;

root->level\_order(root);

}

Question 3

#include <iostream>

using namespace std;

template <typename t>

class queue

{

public:

t data;

queue \*next;

queue(t d)

{

this->next = NULL;

this->data = d;

}

void push(queue \*&front, queue \*&rear, t d)

{

queue \*n = new queue(d);

if (front == NULL)

{

front = n;

rear = n;

return;

}

else

{

rear->next = n;

rear = rear->next;

}

}

void pop(queue \*&front, queue \*&rear)

{

if (front == rear)

{

front = rear = NULL;

}

else

{

queue \*to\_del = front;

front = front->next;

delete (to\_del);

}

}

bool isempty(queue \*front)

{

return front == NULL ? true : false;

}

t front(queue \*&front)

{

if (front != NULL)

{

return front->data;

}

}

};

// bst

class bst

{

public:

int data;

bst \*left;

bst \*right;

bst(int d)

{

this->data = d;

this->left = NULL;

this->right = NULL;

}

bst \*insert\_into\_bst(bst \*root, int d)

{

if (root == NULL)

{

bst \*n = new bst(d);

return n;

}

if (root->data < d)

{

root->right = insert\_into\_bst(root->right, d);

}

else

{

root->left = insert\_into\_bst(root->left, d);

}

return root;

}

void create(bst \*&root)

{

cout << " enter the data : (0 to exit ) : ";

int d;

cin >> d;

while (d)

{

root = root->insert\_into\_bst(root, d);

cout << " enter the data : (0 to exit ) : ";

cin >> d;

}

}

void inorder(bst \*root)

{

if (root == NULL)

return;

inorder(root->left);

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

inorder(root->right);

}

void search(bst \* root ,bst\* & parent ,bool & flag,int key)

{

if(root==NULL)

{

return;

}

if(root->data==key)

{

flag=true;

return;

}

else if(root->left!=NULL&&root->left->data==key)

{

flag=true;

parent=root;

return ;

}

else if(root->right!=NULL&&root->right->data==key)

{

flag=true;

parent=root;

return;

}

search(root->left,parent,flag,key);

search(root->right,parent,flag,key);

}

void level\_order(bst \*root)

{

queue<bst \*> \*qf = NULL;

queue<bst \*> \*qr = NULL;

qf->push(qf, qr, root);

qf->push(qf, qr, NULL);

while (!qf->isempty(qf))

{

bst \*front = qf->front(qf);

qf->pop(qf, qr);

if (front == NULL)

{

if (!qf->isempty(qf))

{

qf->push(qf, qr, NULL);

}

cout << endl;

}

else

{

cout<<front->data<<" ";

if (front->left)

{

qf->push(qf, qr, front->left);

}

if (front->right)

{

qf->push(qf, qr, front->right);

}

}

}

}

};

int main()

{

bst \*root = NULL;

root->create(root);

cout<<"\n enter data to search : ";

int d;

cin>>d;

bool flag=false;

bst \* parent =NULL;

root->search(root,parent,flag,d);

if(flag)

{

if(parent==NULL)

{

cout<<"root is key :";

}

else

{

cout<<"parent : "<<parent->data;

}

}

else

{

cout<<"data is not found ";

}

}

Question 4

#include <iostream>

#include<vector>

using namespace std;

template <typename t>

class queue

{

public:

t data;

queue \*next;

queue(t d)

{

this->next = NULL;

this->data = d;

}

void push(queue \*&front, queue \*&rear, t d)

{

queue \*n = new queue(d);

if (front == NULL)

{

front = n;

rear = n;

return;

}

else

{

rear->next = n;

rear = rear->next;

}

}

void pop(queue \*&front, queue \*&rear)

{

if (front == rear)

{

front = rear = NULL;

}

else

{

queue \*to\_del = front;

front = front->next;

delete (to\_del);

}

}

bool isempty(queue \*front)

{

return front == NULL ? true : false;

}

t front(queue \*&front)

{

if (front != NULL)

{

return front->data;

}

}

};

// bst

class bst

{

public:

int data;

bst \*left;

bst \*right;

bst(int d)

{

this->data = d;

this->left = NULL;

this->right = NULL;

}

bst \*insert\_into\_bst(bst \*root, int d)

{

if (root == NULL)

{

bst \*n = new bst(d);

return n;

}

if (root->data < d)

{

root->right = insert\_into\_bst(root->right, d);

}

else

{

root->left = insert\_into\_bst(root->left, d);

}

return root;

}

void create(bst \*&root)

{

cout << " enter the data : (0 to exit ) : ";

int d;

cin >> d;

while (d)

{

root = root->insert\_into\_bst(root, d);

cout << " enter the data : (0 to exit ) : ";

cin >> d;

}

}

void inorder(bst \*root,vector<int> & a)

{

if (root == NULL)

return;

inorder(root->left,a);

a.push\_back(root->data);

inorder(root->right,a);

}

void level\_order(bst \*root)

{

queue<bst \*> \*qf = NULL;

queue<bst \*> \*qr = NULL;

qf->push(qf, qr, root);

qf->push(qf, qr, NULL);

while (!qf->isempty(qf))

{

bst \*front = qf->front(qf);

qf->pop(qf, qr);

if (front == NULL)

{

if (!qf->isempty(qf))

{

qf->push(qf, qr, NULL);

}

cout << endl;

}

else

{

cout<<front->data<<" ";

if (front->left)

{

qf->push(qf, qr, front->left);

}

if (front->right)

{

qf->push(qf, qr, front->right);

}

}

}

}

bool check(vector<int> & a)

{

for(int i=0;i<a.size()-1;i++)

{

if(a[i]>a[i+1]) return false;

}

return true;

}

};

int main()

{

bst \*root = NULL;

root->create(root);

vector<int> ans;

root->inorder(root,ans);

if(root->check(ans))

{

cout<<"its a bst ";

}

else

{

cout<<"its not a bst ";

}

}

Question 5

#include <iostream>

using namespace std;

template <typename t>

class queue

{

public:

t data;

queue \*next;

queue(t d)

{

this->next = NULL;

this->data = d;

}

void push(queue \*&front, queue \*&rear, t d)

{

queue \*n = new queue(d);

if (front == NULL)

{

front = n;

rear = n;

return;

}

else

{

rear->next = n;

rear = rear->next;

}

}

void pop(queue \*&front, queue \*&rear)

{

if (front == rear)

{

front = rear = NULL;

}

else

{

queue \*to\_del = front;

front = front->next;

delete (to\_del);

}

}

bool isempty(queue \*front)

{

return front == NULL ? true : false;

}

t front(queue \*&front)

{

if (front != NULL)

{

return front->data;

}

}

};

// bst

class bst

{

public:

int data;

bst \*left;

bst \*right;

bst(int d)

{

this->data = d;

this->left = NULL;

this->right = NULL;

}

bst \*insert\_into\_bst(bst \*root, int d)

{

if (root == NULL)

{

bst \*n = new bst(d);

return n;

}

if (root->data < d)

{

root->right = insert\_into\_bst(root->right, d);

}

else

{

root->left = insert\_into\_bst(root->left, d);

}

return root;

}

void create(bst \*&root)

{

cout << " enter the data : (0 to exit ) : ";

int d;

cin >> d;

while (d)

{

root = root->insert\_into\_bst(root, d);

cout << " enter the data : (0 to exit ) : ";

cin >> d;

}

}

void inorder(bst \*root)

{

if (root == NULL)

return;

inorder(root->left);

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

inorder(root->right);

}

bst \* minval(bst\*root)

{

bst \* temp=root;

while(temp->left!=NULL)

{

temp=temp->left;

}

return temp;

}

bst \* del(bst \* root,int key)

{

if(root==NULL) return root;

else if(root->data<key)

{

root->right=del(root->right,key);

}

else if(root->data>key)

{

root->left=del(root->left,key);

}

else

{

if(root->left==NULL)

{

bst \*temp=root->right;

delete root;

return temp;

}

else if(root->right==NULL)

{

bst \*temp=root->left;

delete root;

return temp;

}

else

{

bst \* temp=minval(root->right);

root->data=temp->data;

root->right=del(root->right,temp->data);

}

}

return root;

}

void level\_order(bst \*root)

{

queue<bst \*> \*qf = NULL;

queue<bst \*> \*qr = NULL;

qf->push(qf, qr, root);

qf->push(qf, qr, NULL);

while (!qf->isempty(qf))

{

bst \*front = qf->front(qf);

qf->pop(qf, qr);

if (front == NULL)

{

if (!qf->isempty(qf))

{

qf->push(qf, qr, NULL);

}

cout << endl;

}

else

{

cout<<front->data<<" ";

if (front->left)

{

qf->push(qf, qr, front->left);

}

if (front->right)

{

qf->push(qf, qr, front->right);

}

}

}

}

};

int main()

{

bst \*root = NULL;

root->create(root);

int d;

cout<<"enter the data to delete : ";

cin>>d;

root->del(root,d);

cout<<endl;

root->inorder(root);

}