Jiachen Liu

Problem 1:

a.

The tree look like:

50

20 60

10 40 70

30 45 65 80

25 35 75

b.

After deleting, the tree looks like:

50

25 60

10 40 70

35 45 65 80

75

c.

In order:10-20-25-30-35-40-45-50-60-65-70-75-80

Pre order: 50-20-10-40-30-25-35-45-60-70-65-80-75

Post order: 10-25-35-30-45-40-20-65-75-80-70-60-50

Problem 2:

a.

6

3 5

1 2 4

b.

the array is : 6-3-5-1-2-4

c. after deleting, the array is :5-3-4-1-2

Problem 3:

a.

class Node

{

Node \*left;

Node \*right;

Node \*parent;

int m\_value;

};

b.

//insert

void function(Node\*& n, Node \*parent, int data)

{

if (the current node n is NULL )

create a new node with appropriate constructor and assigns the value. Set the parent pointer of the Node to parent, set both children to NULL.

else if(data is larger than the current node n’s value)

recursively call (n->right, data, n)

else (data is smaller than the current node’s value)

recursively call (n->left, data, n)

}

when we insert we call

function(root, NULL, value)

problem 4:

a.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | A | B | C | D | E | F | G | H | I |
| A | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| B | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| C | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| D | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| E | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| F | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| G | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| H | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| I | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

adjacency list:

|  |  |  |
| --- | --- | --- |
| A | adjacent to | B, C |
| B | adjacent to | H |
| C | adjacent to | D,E |
| D | adjacent to | B,H |
| E | adjacent to | G |
| F | adjacent to | G,I |
| G | adjacent to | C,H |
| H | adjacent to | G |
| I | adjacent to | C |
|  |  |  |

b.

depth-first search:

E-G-C-D-B-H

E-G-C-D-H B

E-G-H C-D-B

(no ‘-’ means interrupted connected)