Contents

Chapter 1

Rubric

Question	Points
Question 1	10
Question 2	10
Question 3	10
Question 4	10
Question 5	10
`rotateLeft`	
Test Cases	1×15
Compilation	10
`rotateLeft` Total	25
`rotateRight`	
Test Cases	1×15
Compilation	10
`rotateRight` Total	25
Total	100

Chapter 2

Metadata

2.1 Submitted Files

handin.time

```
12/05/2019 12:25:23 fsandhu: OurCSCE310Tree.cpp  - 1 day late
12/05/2019 12:25:24 fsandhu: OurCSCE310Tree.h    - 1 day late
12/05/2019 12:34:31 fsandhu: OurCSCE310Tree.cpp  - 1 day late
12/05/2019 12:37:38 fsandhu: OurCSCE310Tree.cpp  - 1 day late
12/05/2019 14:35:52 fsandhu: OurCSCE310Tree.cpp  - 1 day late
12/09/2019 16:27:07 fsandhu: OurCSCE310Tree.cpp  - 5 days late
12/09/2019 16:30:05 fsandhu: OurCSCE310Tree.cpp  - 5 days late
12/09/2019 19:02:52 fsandhu: fsandhu_hw05.pdf    - 5 days late
12/09/2019 19:02:55 fsandhu: OurCSCE310Tree.cpp  - 5 days late
```

2.2 webgrader Runs

webgrader.time

```
2019-12-05T12:25:31-0600 10.43.86.40  fsandhu 005
2019-12-05T12:34:39-0600 10.43.86.40  fsandhu 005
2019-12-05T12:34:55-0600 10.43.86.40  fsandhu 005
2019-12-05T12:37:45-0600 10.43.86.40  fsandhu 005
2019-12-05T14:31:24-0600 10.43.86.40  fsandhu 005
2019-12-05T14:35:55-0600 10.43.86.40  fsandhu 005
2019-12-05T14:36:16-0600 10.43.86.40  fsandhu 005
2019-12-09T16:27:15-0600 10.43.32.151  fsandhu 005
2019-12-09T16:30:12-0600 10.43.32.151  fsandhu 005
2019-12-09T19:03:06-0600 10.43.32.151  fsandhu 005
2019-12-10T19:02:29-0600 76.84.50.181  fsandhu 005
2019-12-15T19:54:19-0600 76.84.219.87  fsandhu 005
```

2.3 diffs

submission.diffs

```
csce310h0mework05/fsandhu/03/part03test01.diff
csce310h0mework05/fsandhu/03/part03test03.diff
csce310h0mework05/fsandhu/03/part03test04.diff
csce310h0mework05/fsandhu/03/part03test08.diff
csce310h0mework05/fsandhu/03/part03test09.diff
csce310h0mework05/fsandhu/03/part03test12.diff
csce310h0mework05/fsandhu/03/part03test13.diff
```

Chapter 3

Written Exercises

CSCE 310 Assignment 5

Fateh Sandhu (17286643)

Q1).

b).

three coins

ⓒ ⓐ ⓑ

First we compare c_1 & c_2

three cases arise:

1). if c_1 & c_2 have same weight:

compare either of the two coins with c_3 to check if all are genuine or c_3 is fake

2) if c_1 is heavier than c_2:

then compare weights of c_1 & c_2 with c_3 to find out which one if fake

3) if c_2 is heavier than c_1:

← Similarly check c_2 & c_1 with c_3 to find out fake coin

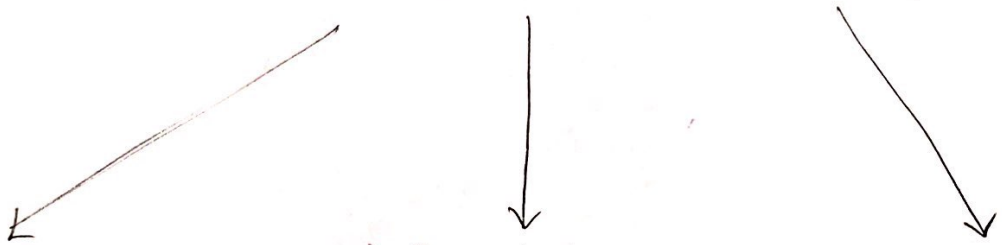
c) Since we don't know if the fake coin is heavier or lighter than the actual genuine coin, we cannot figure out which pile of two coins has the fake coin, we would need at least 4 comparisons to figure out the fake coin.

d).

now we have 4+1 coins . lets call the genuine coin G and others C_1, C_2, C_3, C_4

Replace one coin with G, say C_2

now we weigh G, C_1 and C_3, C_4.



if both piles are equal

C_2 is weighed with G to see if it's fake or not

G, C_1 is heavier:

Compare C_3 and C_4.
if they are equal,
C_1 is fake & heavy
if not, lighter coin (C_3, C_4) is fake

C_3, C_4 is heavier:

compare C_3 & C_4
if C_3, C_4 are equal
then C_1 is fake & light
if not, heavier coin (C_3, C_4) is fake

e) divide 12 coins into 3×4 piles

① compare any two piles

if both have equal weight, the 3'rd pile has the fake coin, if not, the heavier pile has the fake coin

② use the pile and now divide it in two piles of two

heavier pile of 2 has the fake coin.

③ standard comparison b/w 2 coins to find the fake coin

(02)

Merge sort

each subsequence of k elements will require $k \log k$ comparisons to be sorted using any of the $n \log n$ algorithms.

each subsequence will require $k \log k$ comparisons we have a total of n/k subsequences.

$$\text{total comparisons} = \frac{n}{k} \cdot k \log k$$

$$= n \log k$$

$\Omega(n \log k)$ because each subsequence has $\Omega(k \log k)$.

(Q3). $n^{\log_2 n}$ algorithm grows to very large values as we increase the input. $n^{\log_2 n} = n^3$ when $n = 8$ but after that $n^{\log_2 n}$ has a much larger number of comparisons than n^3.

It is intractable.

This can also be verified by graphing n^3 and $n^{\log_2 n}$.

$$n^{\log_2 n} < n^n \quad \text{when } n > 1.$$

$c = 1$.

It becomes a $O(n^n)$.

Q4)

class P has problems solvable in polynomial time.

the brute force algorithm has $\lfloor \frac{n}{2} \rfloor$ comparison for input n so it will be O(n)

but the algorithm actually has way more comparison because it uses binary expansions and bits of input n.

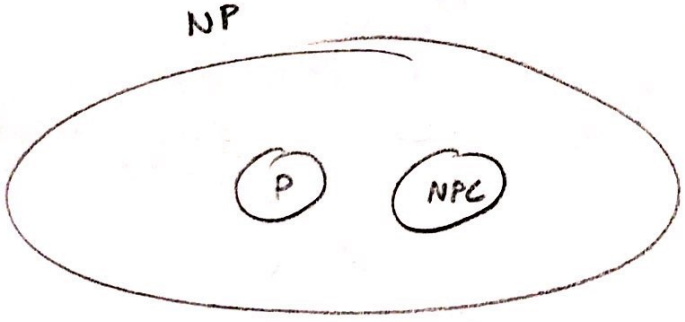
Time changes to $O(2^{\text{bits of } n})$ which changes the algorithm to exponential so it will not be in class P.

Contradictions:

a). A P problem cannot be NP

b). P ≠ NP

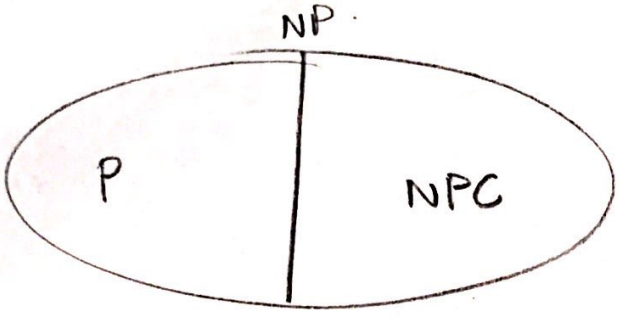
Q5) e does not contradict.



d). every P problem is also checkable in P and NoNPC can belong to P

P and NPC are both solvable or either Checkable in Non deterministic polynomial (NP) time

c does not also contradict



It is the same as e) but e has some extra problems which are neither NPC and P but our knowledge until now suggest c and e.

Chapter 4

Programming Exercises

4.1 rotateLeft

4.1.1 Test 01

diff

part01test01.diff

Input

part01test01.adds.input

```
14
15
18
21
22
25
30
35
```

Submission Output

part01test01.output

```
21,15,14,18,25,22,30,35
14,15,18,21,22,25,30,35
14,18,15,22,35,30,25,21
```

Solution Output

part01test01.solution

```
21,15,14,18,25,22,30,35
14,15,18,21,22,25,30,35
14,18,15,22,35,30,25,21
```

stderr

part01test01.err

4.1.2 Test 02

diff

part01test02.diff

Input

part01test02.adds.input

```
22
24
27
30
34
```

Submission Output

part01test02.output

```
24,22,30,27,34
22,24,27,30,34
22,27,34,30,24
```

Solution Output

part01test02.solution

```
24,22,30,27,34
22,24,27,30,34
22,27,34,30,24
```

stderr

part01test02.err

4.1.3 Test 03

diff

part01test03.diff

Input

part01test03.adds.input

```
15
20
21
23
25
26
29
35
36
39
```

Submission Output

part01test03.output

```
23,20,15,21,35,26,25,29,36,39
15,20,21,23,25,26,29,35,36,39
15,21,20,25,29,26,39,36,35,23
```

Solution Output

part01test03.solution

```
23,20,15,21,35,26,25,29,36,39
15,20,21,23,25,26,29,35,36,39
15,21,20,25,29,26,39,36,35,23
```

stderr

part01test03.err

4.1.4 Test 04

diff

part01test04.diff

Input

part01test04.adds.input

```
22
36
39
```

Submission Output

part01test04.output

```
36,22,39
22,36,39
22,39,36
```

Solution Output

part01test04.solution

```
36,22,39
22,36,39
22,39,36
```

stderr

part01test04.err

4.1.5 Test 05

diff

part01test05.diff

Input

part01test05.adds.input

```
13
14
18
35
37
39
```

Submission Output

part01test05.output

```
35,14,13,18,37,39
13,14,18,35,37,39
13,18,14,39,37,35
```

Solution Output

part01test05.solution

```
35,14,13,18,37,39
13,14,18,35,37,39
13,18,14,39,37,35
```

stderr

part01test05.err

4.1.6 Test 06

diff

part01test06.diff

Input

part01test06.adds.input

```
19
22
25
27
28
31
32
33
35
37
```

Submission Output

part01test06.output

```
27,22,19,25,33,31,28,32,35,37
19,22,25,27,28,31,32,33,35,37
19,25,22,28,32,31,37,35,33,27
```

Solution Output

part01test06.solution

```
27,22,19,25,33,31,28,32,35,37
19,22,25,27,28,31,32,33,35,37
19,25,22,28,32,31,37,35,33,27
```

stderr

part01test06.err

4.1.7 Test 07

diff

part01test07.diff

Input

part01test07.adds.input

```
22
24
26
36
37
```

Submission Output

part01test07.output

```
24,22,36,26,37
22,24,26,36,37
22,26,37,36,24
```

Solution Output

part01test07.solution

```
24,22,36,26,37
22,24,26,36,37
22,26,37,36,24
```

```
stderr
```

part01test07.err

4.1.8 Test 08

```
diff
```

part01test08.diff

Input

part01test08.adds.input

```
17
22
24
33
```

Submission Output

part01test08.output

```
22,17,24,33
17,22,24,33
17,33,24,22
```

Solution Output

part01test08.solution

```
22,17,24,33
17,22,24,33
17,33,24,22
```

```
stderr
```

part01test08.err

4.1.9 Test 09

```
diff
```

part01test09.diff

Input

part01test09.adds.input

```
15
28
29
39
```

Submission Output

part01test09.output

```
28,15,29,39
15,28,29,39
15,39,29,28
```

Solution Output

part01test09.solution

```
28,15,29,39
15,28,29,39
15,39,29,28
```

stderr

part01test09.err

4.1.10 Test 10

diff

part01test10.diff

Input

part01test10.adds.input

```
15
```

Submission Output

part01test10.output

```
15
15
15
```

Solution Output

part01test10.solution

```
15
15
15
```

stderr

part01test10.err

4.1.11 Test 11

diff

part01test11.diff

Input

part01test11.adds.input

```
12
16
23
27
28
29
```

Submission Output

part01test11.output

```
27,16,12,23,28,29
12,16,23,27,28,29
12,23,16,29,28,27
```

Solution Output

part01test11.solution

```
27,16,12,23,28,29
12,16,23,27,28,29
12,23,16,29,28,27
```

stderr

part01test11.err

4.1.12 Test 12

diff

part01test12.diff

Input

part01test12.adds.input

```
27
```

Submission Output

part01test12.output

```
27
27
27
```

Solution Output

part01test12.solution

```
27
27
27
```

stderr

part01test12.err

4.1.13 Test 13

diff

part01test13.diff

Input

part01test13.adds.input

```
12
20
23
24
31
34
```

Submission Output

part01test13.output

```
24,20,12,23,31,34
12,20,23,24,31,34
12,23,20,34,31,24
```

Solution Output

part01test13.solution

```
24,20,12,23,31,34
12,20,23,24,31,34
12,23,20,34,31,24
```

stderr

part01test13.err

4.1.14 Test 14

diff

part01test14.diff

Input

part01test14.adds.input

```
13
20
24
29
35
36
37
38
40
```

Submission Output

part01test14.output

```
29,20,13,24,36,35,38,37,40
13,20,24,29,35,36,37,38,40
13,24,20,35,37,40,38,36,29
```

Solution Output

part01test14.solution

```
29,20,13,24,36,35,38,37,40
13,20,24,29,35,36,37,38,40
13,24,20,35,37,40,38,36,29
```

stderr

part01test14.err

4.1.15 Test 15

diff

part01test15.diff

Input

part01test15.adds.input

```
20
28
30
35
39
```

Submission Output

part01test15.output

```
28,20,35,30,39
20,28,30,35,39
20,30,39,35,28
```

Solution Output

part01test15.solution

```
28,20,35,30,39
20,28,30,35,39
20,30,39,35,28
```

stderr

part01test15.err

4.1.16 Source Code

OurCSCE310Tree.h

```
1  #ifndef OURCSCE310TREE_H
2  #define OURCSCE310TREE_H
3  using namespace std;
4
5  class OurCSCE310Tree{
6   public:
7    OurCSCE310Tree(void);
8    OurCSCE310Tree(OurCSCE310Tree&);
9    ~OurCSCE310Tree(void);
10   void operator=(OurCSCE310Tree&);
11   OurCSCE310Tree* getParent(void);
12   OurCSCE310Tree* getLeft(void);
13   OurCSCE310Tree* getRight(void);
14   int getValue(void);
15   void setParent(OurCSCE310Tree*);
16   void setLeft(OurCSCE310Tree*);
17   void setRight(OurCSCE310Tree*);
18   void setValue(int);
19   void insert(int);
20   void printPreorder(void);
21   void printInorder(void);
22   void printPostorder(void);
23   void rotateLeft(void);
24   void rotateRight(void);
25   void rotateLeftRight(void);
26   void rotateRightLeft(void);
27   void deleteNode(int);
28   int getHeight();
29
30   private:
```

```
31    int value;
32    OurCSCE310Tree* parent;
33    OurCSCE310Tree* left;
34    OurCSCE310Tree* right;
35  };
36
37  #endif
```

OurCSCE310Tree.cpp

```
1   #include "OurCSCE310Tree.h"
2   #include <iostream>
3   #include <cmath>
4   using namespace std;
5
6   /*
7   class OurCSCE310Tree{
8    public:
9     OurCSCE310Tree(void);
10    OurCSCE310Tree(OurCSCE310Tree&);
11    ~OurCSCE310Tree(void);
12    void operator=(OurCSCE310Tree&);
13    OurCSCE310Tree* getParent(void);
14    OurCSCE310Tree* getLeft(void);
15    OurCSCE310Tree* getRight(void);
16    int getValue(void);
17    void setParent(OurCSCE310Tree*);
18    void setLeft(OurCSCE310Tree*);
19    void setRight(OurCSCE310Tree*);
20    void setValue(int);
21    void insert(int);
22    void printPreorder(void);
23    void printInorder(void);
24    void printPostorder(void);
25    void rotateLeft(void);
26    void rotateRight(void);
27    void rotateLeftRight(void);
28    void rotateRightLeft(void);
29    void deleteNode(int);
30    int getHeight();
31
32   private:
33    int value;
34    OurCSCE310Tree* parent;
35    OurCSCE310Tree* left;
36    OurCSCE310Tree* right;
37  };
38   */
39
40  OurCSCE310Tree::OurCSCE310Tree(){
41    value = 0;
42    parent = NULL;
43    left = NULL;
44    right = NULL;
45  }
46
47  OurCSCE310Tree::OurCSCE310Tree( OurCSCE310Tree& other){
48    delete parent;
49    delete left;
50    delete right;
51    value = other.getValue();
```

```
52    parent = other.getParent();
53    left = other.getLeft();
54    right = other.getRight();
55  }
56
57  void OurCSCE310Tree::operator=( OurCSCE310Tree& other){
58    delete parent;
59    delete left;
60    delete right;
61    value = other.getValue();
62    parent = other.getParent();
63    left = other.getLeft();
64    right = other.getRight();
65  }
66
67  OurCSCE310Tree::~OurCSCE310Tree(){
68    delete left;
69    left = NULL;
70    delete right;
71    right = NULL;
72    value = 0;
73  }
74
75  OurCSCE310Tree* OurCSCE310Tree::getParent(){
76    return parent;
77  }
78
79  OurCSCE310Tree* OurCSCE310Tree::getLeft(){
80    return left;
81  }
82
83  OurCSCE310Tree* OurCSCE310Tree::getRight(){
84    return right;
85  }
86
87  int OurCSCE310Tree::getValue(){
88    return value;
89  }
90
91  void OurCSCE310Tree::setParent( OurCSCE310Tree* par ){
92    parent = par;
93  }
94
95  void OurCSCE310Tree::setLeft( OurCSCE310Tree* lft ){
96    left = lft;
97  }
98
99  void OurCSCE310Tree::setRight( OurCSCE310Tree* rght ){
100   right = rght;
101 }
102
103 void OurCSCE310Tree::setValue( int val ){
104   value = val;
105 }
106
107 void OurCSCE310Tree::insert( int val ){
108   if( !getValue() ){
109     setValue( val );
110   }
111   else if( ( val < getValue() && !getLeft() ) || ( val < getValue() && !getLeft()->
```

```
     getValue() ) ){
112      left = new OurCSCE310Tree();
113      left->setParent( this );
114      left->setValue( val );
115    }
116    else if( ( val > getValue() && !getRight() ) || ( val > getValue() && !getRight()->
       getValue() ) ){
117      right = new OurCSCE310Tree();
118      right->setParent( this );
119      right->setValue( val );
120    }
121    else if( val < getValue() ){
122      getLeft()->insert( val );
123    }
124    else{
125      getRight()->insert( val );
126    }
127
128    if( getLeft() && getLeft()->getRight() && !getRight() || getLeft() && getLeft()->
       getRight() && getRight() && getLeft()->getHeight() > getRight()->getHeight() + 1
       && getLeft()->getRight()->getHeight() > getLeft()->getLeft()->getHeight() + 1 ){
129      rotateLeftRight();
130    }
131    else if( getRight() && getRight()->getLeft() && !getLeft() || getRight() && getRight
       ()->getLeft() && getLeft() && getRight()->getHeight() > getLeft()->getHeight() + 1
        && getRight()->getLeft()->getHeight() > getRight()->getRight()->getHeight() + 1 )
       {
132      rotateRightLeft();
133    }
134    else if( getLeft() && !getRight() && getLeft()->getHeight() > 1 || getLeft() &&
       getRight() && getLeft()->getHeight() > getRight()->getHeight() + 1 ){
135      rotateRight();
136    }
137    else if( getRight() && !getLeft() && getRight()->getHeight() > 1 || getRight() &&
       getLeft() && getRight()->getHeight() > getLeft()->getHeight() + 1 ){
138      rotateLeft();
139    }
140  }
141
142  void OurCSCE310Tree::printPreorder(){
143    if( getValue() ){
144      cout << getValue();
145    }
146    if( getLeft() && getLeft()->getValue() ){
147      cout << ",";
148      getLeft()->printPreorder();
149    }
150    if( getRight() && getRight()->getValue() ){
151      cout << ",";
152      getRight()->printPreorder();
153    }
154  }
155
156  void OurCSCE310Tree::printInorder(){
157    if( getLeft() && getLeft()->getValue() ){
158      getLeft()->printInorder();
159      cout << ",";
160    }
161    if( getValue() ){
162      cout << getValue();
```

```
163   }
164   if( getRight() && getRight()->getValue() ){
165     cout << ",";
166     getRight()->printInorder();
167   }
168 }
169
170 void OurCSCE310Tree::printPostorder(){
171   if( getLeft() && getLeft()->getValue() ){
172     getLeft()->printPostorder();
173     cout << ",";
174   }
175   if( getRight() && getRight()->getValue() ){
176     getRight()->printPostorder();
177     cout << ",";
178   }
179   if( getValue() ){
180     cout << getValue();
181   }
182 }
183
184 int OurCSCE310Tree::getHeight(){
185   if( getLeft() && getLeft()->getValue() && ( !getRight() || !getRight()->getValue() )
      ){
186     return getLeft()->getHeight() + 1;
187   }
188   else if( getRight() && getRight()->getValue() && ( !getLeft() || !getLeft()->
     getValue() ) ){
189     return getRight()->getHeight() + 1;
190   }
191   else if( getRight() && getLeft() && getRight()->getValue() && getLeft()->getValue()
     ){
192     return fmax( getRight()->getHeight() , getLeft()->getHeight() ) + 1;
193   }
194   else if( getValue() && ( !getLeft() || !getLeft()->getValue() ) && ( !getRight() ||
     !getRight()->getValue() ) ){
195     return 1;
196   }
197   return 0;
198 }
199
200 void OurCSCE310Tree::rotateLeftRight(){
201   getLeft()->rotateLeft();
202   rotateRight();
203 }
204
205 void OurCSCE310Tree::rotateRightLeft(){
206   getRight()->rotateRight();
207   rotateLeft();
208 }
209
210 void OurCSCE310Tree::rotateLeft(){
211
212   int tempRight = this->getRight()->getValue();
213   OurCSCE310Tree* rightNode = this->getRight();
214   OurCSCE310Tree* leftNode = this->getLeft();
215
216   this->getRight()->setValue(this->getValue());
217   this->setValue(tempRight);
218
```

```
219   this->setRight(this->getRight()->getRight());
220
221   this->setLeft(rightNode);
222   this->getLeft()->setRight(rightNode->getLeft());
223   this->getLeft()->setLeft(leftNode);
224
225 }
226
227 void OurCSCE310Tree::rotateRight(){
228
229   int tempLeft = this->getLeft()->getValue();
230   OurCSCE310Tree* leftNode = this->getLeft();
231   OurCSCE310Tree* rightNode = this->getRight();
232
233   this->getLeft()->setValue(this->getValue());
234   this->setValue(tempLeft);
235
236   this->setLeft(this->getLeft()->getLeft());
237
238   this->setRight(leftNode);
239   this->getRight()->setLeft(leftNode->getRight());
240   this->getRight()->setRight(rightNode);
241
242 }
243
244 void OurCSCE310Tree::deleteNode( int key ){
245
246   //delete root
247   OurCSCE310Tree* node = this;
248   if (node->getValue() == key) {
249     while (node->getLeft() != NULL) {
250       node = node->getLeft();
251     }
252     this->setValue(node->getValue());
253   }
254
255 }
```

4.2 rotateRight

4.2.1 Test 01

diff

part02test01.diff

Input

part02test01.adds.input

```
35
30
27
23
19
12
```

Submission Output

part02test01.output

```
23,19,12,30,27,35
```

```
12,19,23,27,30,35
12,19,27,35,30,23
```

Solution Output

part02test01.solution

```
23,19,12,30,27,35
12,19,23,27,30,35
12,19,27,35,30,23
```

stderr

part02test01.err

4.2.2 Test 02

diff

part02test02.diff

Input

part02test02.adds.input

```
36
32
29
27
26
22
21
19
17
14
```

Submission Output

part02test02.output

```
27,19,17,14,22,21,26,32,29,36
14,17,19,21,22,26,27,29,32,36
14,17,21,26,22,19,29,36,32,27
```

Solution Output

part02test02.solution

```
27,19,17,14,22,21,26,32,29,36
14,17,19,21,22,26,27,29,32,36
14,17,21,26,22,19,29,36,32,27
```

stderr

part02test02.err

4.2.3 Test 03

diff

part02test03.diff

Input

part02test03.adds.input

```
40
36
33
```

Submission Output

part02test03.output

```
36,33,40
33,36,40
33,40,36
```

Solution Output

part02test03.solution

```
36,33,40
33,36,40
33,40,36
```

```
stderr
```

part02test03.err

4.2.4 Test 04

```
diff
```

part02test04.diff

Input

part02test04.adds.input

```
40
37
36
34
32
30
25
24
18
15
```

Submission Output

part02test04.output

```
34,24,18,15,30,25,32,37,36,40
15,18,24,25,30,32,34,36,37,40
15,18,25,32,30,24,36,40,37,34
```

Solution Output

part02test04.solution

```
34,24,18,15,30,25,32,37,36,40
15,18,24,25,30,32,34,36,37,40
15,18,25,32,30,24,36,40,37,34
```

```
stderr
```

part02test04.err

4.2.5 Test 05

diff

part02test05.diff

Input

part02test05.adds.input

```
39
38
35
34
30
28
26
23
21
```

Submission Output

part02test05.output

```
34,28,23,21,26,30,38,35,39
21,23,26,28,30,34,35,38,39
21,26,23,30,28,35,39,38,34
```

Solution Output

part02test05.solution

```
34,28,23,21,26,30,38,35,39
21,23,26,28,30,34,35,38,39
21,26,23,30,28,35,39,38,34
```

stderr

part02test05.err

4.2.6 Test 06

diff

part02test06.diff

Input

part02test06.adds.input

```
36
30
27
25
23
21
17
15
```

Submission Output

part02test06.output

```
25,21,17,15,23,30,27,36
15,17,21,23,25,27,30,36
15,17,23,21,27,36,30,25
```

Solution Output

part02test06.solution

```
25,21,17,15,23,30,27,36
15,17,21,23,25,27,30,36
15,17,23,21,27,36,30,25
```

stderr

part02test06.err

4.2.7 Test 07

diff

part02test07.diff

Input

part02test07.adds.input

```
36
35
29
25
21
20
19
```

Submission Output

part02test07.output

```
25,20,19,21,35,29,36
19,20,21,25,29,35,36
19,21,20,29,36,35,25
```

Solution Output

part02test07.solution

```
25,20,19,21,35,29,36
19,20,21,25,29,35,36
19,21,20,29,36,35,25
```

stderr

part02test07.err

4.2.8 Test 08

diff

part02test08.diff

Input

part02test08.adds.input

```
37
36
31
29
26
```

```
20
19
17
14
```

Submission Output

part02test08.output

```
29,20,17,14,19,26,36,31,37
14,17,19,20,26,29,31,36,37
14,19,17,26,20,31,37,36,29
```

Solution Output

part02test08.solution

```
29,20,17,14,19,26,36,31,37
14,17,19,20,26,29,31,36,37
14,19,17,26,20,31,37,36,29
```

stderr

part02test08.err

4.2.9 Test 09

diff

part02test09.diff

Input

part02test09.adds.input

```
40
36
34
33
23
16
15
13
```

Submission Output

part02test09.output

```
33,16,15,13,23,36,34,40
13,15,16,23,33,34,36,40
13,15,23,16,34,40,36,33
```

Solution Output

part02test09.solution

```
33,16,15,13,23,36,34,40
13,15,16,23,33,34,36,40
13,15,23,16,34,40,36,33
```

stderr

part02test09.err

4.2.10 Test 10

diff

part02test10.diff

Input

part02test10.adds.input

```
39
33
32
28
27
26
20
19
12
```

Submission Output

part02test10.output

```
28,26,19,12,20,27,33,32,39
12,19,20,26,27,28,32,33,39
12,20,19,27,26,32,39,33,28
```

Solution Output

part02test10.solution

```
28,26,19,12,20,27,33,32,39
12,19,20,26,27,28,32,33,39
12,20,19,27,26,32,39,33,28
```

stderr

part02test10.err

4.2.11 Test 11

diff

part02test11.diff

Input

part02test11.adds.input

```
34
32
31
22
16
```

Submission Output

part02test11.output

```
32,22,16,31,34
16,22,31,32,34
16,31,22,34,32
```

Solution Output

part02test11.solution

```
32,22,16,31,34
16,22,31,32,34
16,31,22,34,32
```

stderr

part02test11.err

4.2.12 Test 12

diff

part02test12.diff

Input

part02test12.adds.input

```
37
31
29
23
21
18
16
```

Submission Output

part02test12.output

```
23,18,16,21,31,29,37
16,18,21,23,29,31,37
16,21,18,29,37,31,23
```

Solution Output

part02test12.solution

```
23,18,16,21,31,29,37
16,18,21,23,29,31,37
16,21,18,29,37,31,23
```

stderr

part02test12.err

4.2.13 Test 13

diff

part02test13.diff

Input

part02test13.adds.input

```
33
```

Submission Output

part02test13.output

```
33
33
33
```

Solution Output

part02test13.solution

```
33
33
33
```

stderr

part02test13.err

4.2.14 Test 14

diff

part02test14.diff

Input

part02test14.adds.input

```
39
32
30
28
27
21
19
18
15
```

Submission Output

part02test14.output

```
28,21,18,15,19,27,32,30,39
15,18,19,21,27,28,30,32,39
15,19,18,27,21,30,39,32,28
```

Solution Output

part02test14.solution

```
28,21,18,15,19,27,32,30,39
15,18,19,21,27,28,30,32,39
15,19,18,27,21,30,39,32,28
```

stderr

part02test14.err

4.2.15 Test 15

diff

part02test15.diff

Input

part02test15.adds.input

```
36
35
18
14
12
```

Submission Output

part02test15.output

```
35,14,12,18,36
12,14,18,35,36
12,18,14,36,35
```

Solution Output

part02test15.solution

```
35,14,12,18,36
12,14,18,35,36
12,18,14,36,35
```

stderr

part02test15.err

4.2.16 Source Code

OurCSCE310Tree.h

```
1  #ifndef OURCSCE310TREE_H
2  #define OURCSCE310TREE_H
3  using namespace std;
4
5  class OurCSCE310Tree{
6   public:
7    OurCSCE310Tree(void);
8    OurCSCE310Tree(OurCSCE310Tree&);
9    ~OurCSCE310Tree(void);
10   void operator=(OurCSCE310Tree&);
11   OurCSCE310Tree* getParent(void);
12   OurCSCE310Tree* getLeft(void);
13   OurCSCE310Tree* getRight(void);
14   int getValue(void);
15   void setParent(OurCSCE310Tree*);
16   void setLeft(OurCSCE310Tree*);
17   void setRight(OurCSCE310Tree*);
18   void setValue(int);
19   void insert(int);
20   void printPreorder(void);
21   void printInorder(void);
22   void printPostorder(void);
23   void rotateLeft(void);
24   void rotateRight(void);
25   void rotateLeftRight(void);
26   void rotateRightLeft(void);
27   void deleteNode(int);
28   int getHeight();
29
30  private:
31   int value;
32   OurCSCE310Tree* parent;
33   OurCSCE310Tree* left;
34   OurCSCE310Tree* right;
35 };
36
37 #endif
```

```
1  #include "OurCSCE310Tree.h"
2  #include <iostream>
3  #include <cmath>
4  using namespace std;
5
6  /*
7  class OurCSCE310Tree{
8   public:
9    OurCSCE310Tree(void);
10   OurCSCE310Tree(OurCSCE310Tree&);
11   ~OurCSCE310Tree(void);
12   void operator=(OurCSCE310Tree&);
13   OurCSCE310Tree* getParent(void);
14   OurCSCE310Tree* getLeft(void);
15   OurCSCE310Tree* getRight(void);
16   int getValue(void);
17   void setParent(OurCSCE310Tree*);
18   void setLeft(OurCSCE310Tree*);
19   void setRight(OurCSCE310Tree*);
20   void setValue(int);
21   void insert(int);
22   void printPreorder(void);
23   void printInorder(void);
24   void printPostorder(void);
25   void rotateLeft(void);
26   void rotateRight(void);
27   void rotateLeftRight(void);
28   void rotateRightLeft(void);
29   void deleteNode(int);
30   int getHeight();
31
32  private:
33   int value;
34   OurCSCE310Tree* parent;
35   OurCSCE310Tree* left;
36   OurCSCE310Tree* right;
37 };
38  */
39
40 OurCSCE310Tree::OurCSCE310Tree(){
41   value = 0;
42   parent = NULL;
43   left = NULL;
44   right = NULL;
45 }
46
47 OurCSCE310Tree::OurCSCE310Tree( OurCSCE310Tree& other){
48   delete parent;
49   delete left;
50   delete right;
51   value = other.getValue();
52   parent = other.getParent();
53   left = other.getLeft();
54   right = other.getRight();
55 }
56
57 void OurCSCE310Tree::operator=( OurCSCE310Tree& other){
58   delete parent;
59   delete left;
```

```
60    delete right;
61    value = other.getValue();
62    parent = other.getParent();
63    left = other.getLeft();
64    right = other.getRight();
65  }
66
67  OurCSCE310Tree::~OurCSCE310Tree(){
68    delete left;
69    left = NULL;
70    delete right;
71    right = NULL;
72    value = 0;
73  }
74
75  OurCSCE310Tree* OurCSCE310Tree::getParent(){
76    return parent;
77  }
78
79  OurCSCE310Tree* OurCSCE310Tree::getLeft(){
80    return left;
81  }
82
83  OurCSCE310Tree* OurCSCE310Tree::getRight(){
84    return right;
85  }
86
87  int OurCSCE310Tree::getValue(){
88    return value;
89  }
90
91  void OurCSCE310Tree::setParent( OurCSCE310Tree* par ){
92    parent = par;
93  }
94
95  void OurCSCE310Tree::setLeft( OurCSCE310Tree* lft ){
96    left = lft;
97  }
98
99  void OurCSCE310Tree::setRight( OurCSCE310Tree* rght ){
100   right = rght;
101 }
102
103 void OurCSCE310Tree::setValue( int val ){
104   value = val;
105 }
106
107 void OurCSCE310Tree::insert( int val ){
108   if( !getValue() ){
109     setValue( val );
110   }
111   else if( ( val < getValue() && !getLeft() ) || ( val < getValue() && !getLeft()->
       getValue() ) ){
112     left = new OurCSCE310Tree();
113     left->setParent( this );
114     left->setValue( val );
115   }
116   else if( ( val > getValue() && !getRight() ) || ( val > getValue() && !getRight()->
       getValue() ) ){
117     right = new OurCSCE310Tree();
```

```
118       right->setParent( this );
119       right->setValue( val );
120     }
121     else if( val < getValue() ){
122       getLeft()->insert( val );
123     }
124     else{
125       getRight()->insert( val );
126     }
127
128     if( getLeft() && getLeft()->getRight() && !getRight() || getLeft() && getLeft()->
        getRight() && getRight() && getLeft()->getHeight() > getRight()->getHeight() + 1
        && getLeft()->getRight()->getHeight() > getLeft()->getLeft()->getHeight() + 1 ){
129       rotateLeftRight();
130     }
131     else if( getRight() && getRight()->getLeft() && !getLeft() || getRight() && getRight
        ()->getLeft() && getLeft() && getRight()->getHeight() > getLeft()->getHeight() + 1
         && getRight()->getLeft()->getHeight() > getRight()->getRight()->getHeight() + 1 )
        {
132       rotateRightLeft();
133     }
134     else if( getLeft() && !getRight() && getLeft()->getHeight() > 1 || getLeft() &&
        getRight() && getLeft()->getHeight() > getRight()->getHeight() + 1 ){
135       rotateRight();
136     }
137     else if( getRight() && !getLeft() && getRight()->getHeight() > 1 || getRight() &&
        getLeft() && getRight()->getHeight() > getLeft()->getHeight() + 1 ){
138       rotateLeft();
139     }
140 }
141
142 void OurCSCE310Tree::printPreorder(){
143     if( getValue() ){
144       cout << getValue();
145     }
146     if( getLeft() && getLeft()->getValue() ){
147       cout << ",";
148       getLeft()->printPreorder();
149     }
150     if( getRight() && getRight()->getValue() ){
151       cout << ",";
152       getRight()->printPreorder();
153     }
154 }
155
156 void OurCSCE310Tree::printInorder(){
157     if( getLeft() && getLeft()->getValue() ){
158       getLeft()->printInorder();
159       cout << ",";
160     }
161     if( getValue() ){
162       cout << getValue();
163     }
164     if( getRight() && getRight()->getValue() ){
165       cout << ",";
166       getRight()->printInorder();
167     }
168 }
169
170 void OurCSCE310Tree::printPostorder(){
```

```
171   if( getLeft() && getLeft()->getValue() ){
172     getLeft()->printPostorder();
173     cout << ",";
174   }
175   if( getRight() && getRight()->getValue() ){
176     getRight()->printPostorder();
177     cout << ",";
178   }
179   if( getValue() ){
180     cout << getValue();
181   }
182 }
183
184 int OurCSCE310Tree::getHeight(){
185   if( getLeft() && getLeft()->getValue() && ( !getRight() || !getRight()->getValue() )
        ){
186     return getLeft()->getHeight() + 1;
187   }
188   else if( getRight() && getRight()->getValue() && ( !getLeft() || !getLeft()->
       getValue() ) ){
189     return getRight()->getHeight() + 1;
190   }
191   else if( getRight() && getLeft() && getRight()->getValue() && getLeft()->getValue()
       ){
192     return fmax( getRight()->getHeight() , getLeft()->getHeight() ) + 1;
193   }
194   else if( getValue() && ( !getLeft() || !getLeft()->getValue() ) && ( !getRight() ||
       !getRight()->getValue() ) ){
195     return 1;
196   }
197   return 0;
198 }
199
200 void OurCSCE310Tree::rotateLeftRight(){
201   getLeft()->rotateLeft();
202   rotateRight();
203 }
204
205 void OurCSCE310Tree::rotateRightLeft(){
206   getRight()->rotateRight();
207   rotateLeft();
208 }
209
210 void OurCSCE310Tree::rotateLeft(){
211
212   int tempRight = this->getRight()->getValue();
213   OurCSCE310Tree* rightNode = this->getRight();
214   OurCSCE310Tree* leftNode = this->getLeft();
215
216   this->getRight()->setValue(this->getValue());
217   this->setValue(tempRight);
218
219   this->setRight(this->getRight()->getRight());
220
221   this->setLeft(rightNode);
222   this->getLeft()->setRight(rightNode->getLeft());
223   this->getLeft()->setLeft(leftNode);
224
225 }
226
```

```
227 void OurCSCE310Tree::rotateRight(){
228
229   int tempLeft = this->getLeft()->getValue();
230   OurCSCE310Tree* leftNode = this->getLeft();
231   OurCSCE310Tree* rightNode = this->getRight();
232
233   this->getLeft()->setValue(this->getValue());
234   this->setValue(tempLeft);
235
236   this->setLeft(this->getLeft()->getLeft());
237
238   this->setRight(leftNode);
239   this->getRight()->setLeft(leftNode->getRight());
240   this->getRight()->setRight(rightNode);
241
242 }
243
244 void OurCSCE310Tree::deleteNode( int key ){
245
246   //delete root
247   OurCSCE310Tree* node = this;
248   if (node->getValue() == key) {
249     while (node->getLeft() != NULL) {
250       node = node->getLeft();
251     }
252     this->setValue(node->getValue());
253   }
254
255 }
```

4.3 deleteNode

4.3.1 Test 01

diff

part03test01.diff

```
1c1
< 31,33,34
---
> 33,34
```

Input

part03test01.adds.input

```
31
34
33
```

part03test01.deletes.input

```
17
13
19
16
26
31
27
```

Submission Output

part03test01.output

```
31,33,34
```

Solution Output

part03test01.solution

```
33,34
```

```
stderr
```

part03test01.err

4.3.2 Test 02

```
diff
```

part03test02.diff

Input

part03test02.adds.input

```
39
13
27
```

part03test02.deletes.input

```
31
34
25
28
36
15
21
```

Submission Output

part03test02.output

```
13,27,39
```

Solution Output

part03test02.solution

```
13,27,39
```

```
stderr
```

part03test02.err

4.3.3 Test 03

```
diff
```

part03test03.diff

```
1c1
< 14,23
---
> 23
```

Input

part03test03.adds.input

```
23
14
```

part03test03.deletes.input

```
33
26
29
25
14
34
32
```

Submission Output

part03test03.output

```
14,23
```

Solution Output

part03test03.solution

```
23
```

```
stderr
```

part03test03.err

4.3.4 Test 04

```
diff
```

part03test04.diff

```
1c1
< 21,30,21,33
---
> 21,30,33
```

Input

part03test04.adds.input

```
32
30
33
21
```

part03test04.deletes.input

```
37
16
20
34
32
31
23
```

Submission Output

part03test04.output

```
21,30,21,33
```

Solution Output

part03test04.solution

```
21,30,33
```

```
stderr
```

part03test04.err

4.3.5 Test 05

```
diff
```

part03test05.diff

Input

part03test05.adds.input

```
34
32
```

part03test05.deletes.input

```
33
18
21
20
12
31
15
28
27
```

Submission Output

part03test05.output

```
32,34
```

Solution Output

part03test05.solution

```
32,34
```

```
stderr
```

part03test05.err

4.3.6 Test 06

```
diff
```

part03test06.diff

Input

part03test06.adds.input

```
40
```

part03test06.deletes.input

```
14
29
25
18
```

```
15
23
27
13
```

Submission Output

part03test06.output

```
40
```

Solution Output

part03test06.solution

```
40
```

```
stderr
```

part03test06.err

4.3.7 Test 07

```
diff
```

part03test07.diff

Input

part03test07.adds.input

```
34
16
24
13
```

part03test07.deletes.input

```
22
32
17
15
31
38
```

Submission Output

part03test07.output

```
13,16,24,34
```

Solution Output

part03test07.solution

```
13,16,24,34
```

```
stderr
```

part03test07.err

4.3.8 Test 08

```
diff
```

part03test08.diff

```
1c1
< 12,12,26
---
> 12,26
```

Input

part03test08.adds.input

```
15
26
12
```

part03test08.deletes.input

```
33
16
30
29
38
24
15
23
37
```

Submission Output

part03test08.output

```
12,12,26
```

Solution Output

part03test08.solution

```
12,26
```

```
stderr
```

part03test08.err

4.3.9 Test 09

```
diff
```

part03test09.diff

```
1c1
< 21,21,31
---
> 21
```

Input

part03test09.adds.input

```
31
21
25
```

part03test09.deletes.input

```
12
28
26
```

```
32
29
31
25
27
16
```

Submission Output

part03test09.output

```
21,21,31
```

Solution Output

part03test09.solution

```
21
```

```
stderr
```

part03test09.err

4.3.10 Test 10

```
diff
```

part03test10.diff

Input

part03test10.adds.input

```
32
26
34
```

part03test10.deletes.input

```
39
28
12
24
16
22
```

Submission Output

part03test10.output

```
26,32,34
```

Solution Output

part03test10.solution

```
26,32,34
```

```
stderr
```

part03test10.err

4.3.11 Test 11

```
diff
```

part03test11.diff

Input

part03test11.adds.input

```
16
37
20
```

part03test11.deletes.input

```
23
22
39
36
34
19
31
21
```

Submission Output

part03test11.output

```
16,20,37
```

Solution Output

part03test11.solution

```
16,20,37
```

```
stderr
```

part03test11.err

4.3.12 Test 12

```
diff
```

part03test12.diff

```
1c1
< 12,24,26,29
---
> 12,24,26
```

Input

part03test12.adds.input

```
29
24
26
12
```

part03test12.deletes.input

```
25
19
33
29
37
40
23
```

Submission Output

part03test12.output

```
12,24,26,29
```

Solution Output

part03test12.solution

```
12,24,26
```

```
stderr
```

part03test12.err

4.3.13 Test 13

```
diff
```

part03test13.diff

```
1c1
< 18,23,25
---
> 18,23
```

Input

part03test13.adds.input

```
25
23
18
```

part03test13.deletes.input

```
16
33
30
21
35
38
19
25
```

Submission Output

part03test13.output

```
18,23,25
```

Solution Output

part03test13.solution

```
18,23
```

```
stderr
```

part03test13.err

4.3.14 Test 14

```
diff
```

part03test14.diff

Input

part03test14.adds.input

```
33
40
```

part03test14.deletes.input

```
27
38
20
18
34
13
14
31
21
```

Submission Output

part03test14.output

```
33,40
```

Solution Output

part03test14.solution

```
33,40
```

`stderr`

part03test14.err

4.3.15 Test 15

`diff`

part03test15.diff

Input

part03test15.adds.input

```
18
12
```

part03test15.deletes.input

```
35
32
28
21
24
30
23
14
```

Submission Output

part03test15.output

```
12,18
```

Solution Output

part03test15.solution

```
12,18
```

```
stderr
```

part03test15.err

4.3.16 Source Code

OurCSCE310Tree.h

```
#ifndef OURCSCE310TREE_H
#define OURCSCE310TREE_H
using namespace std;

class OurCSCE310Tree{
 public:
  OurCSCE310Tree(void);
  OurCSCE310Tree(OurCSCE310Tree&);
  ~OurCSCE310Tree(void);
  void operator=(OurCSCE310Tree&);
  OurCSCE310Tree* getParent(void);
  OurCSCE310Tree* getLeft(void);
  OurCSCE310Tree* getRight(void);
  int getValue(void);
  void setParent(OurCSCE310Tree*);
  void setLeft(OurCSCE310Tree*);
  void setRight(OurCSCE310Tree*);
  void setValue(int);
  void insert(int);
  void printPreorder(void);
  void printInorder(void);
  void printPostorder(void);
  void rotateLeft(void);
  void rotateRight(void);
  void rotateLeftRight(void);
  void rotateRightLeft(void);
  void deleteNode(int);
  int getHeight();

 private:
  int value;
  OurCSCE310Tree* parent;
  OurCSCE310Tree* left;
  OurCSCE310Tree* right;
};

#endif
```

OurCSCE310Tree.cpp

```
#include "OurCSCE310Tree.h"
#include <iostream>
#include <cmath>
using namespace std;

/*
class OurCSCE310Tree{
 public:
  OurCSCE310Tree(void);
  OurCSCE310Tree(OurCSCE310Tree&);
  ~OurCSCE310Tree(void);
  void operator=(OurCSCE310Tree&);
  OurCSCE310Tree* getParent(void);
```

```
14    OurCSCE310Tree* getLeft(void);
15    OurCSCE310Tree* getRight(void);
16    int getValue(void);
17    void setParent(OurCSCE310Tree*);
18    void setLeft(OurCSCE310Tree*);
19    void setRight(OurCSCE310Tree*);
20    void setValue(int);
21    void insert(int);
22    void printPreorder(void);
23    void printInorder(void);
24    void printPostorder(void);
25    void rotateLeft(void);
26    void rotateRight(void);
27    void rotateLeftRight(void);
28    void rotateRightLeft(void);
29    void deleteNode(int);
30    int getHeight();
31
32   private:
33    int value;
34    OurCSCE310Tree* parent;
35    OurCSCE310Tree* left;
36    OurCSCE310Tree* right;
37  };
38   */
39
40  OurCSCE310Tree::OurCSCE310Tree(){
41    value = 0;
42    parent = NULL;
43    left = NULL;
44    right = NULL;
45  }
46
47  OurCSCE310Tree::OurCSCE310Tree( OurCSCE310Tree& other){
48    delete parent;
49    delete left;
50    delete right;
51    value = other.getValue();
52    parent = other.getParent();
53    left = other.getLeft();
54    right = other.getRight();
55  }
56
57  void OurCSCE310Tree::operator=( OurCSCE310Tree& other){
58    delete parent;
59    delete left;
60    delete right;
61    value = other.getValue();
62    parent = other.getParent();
63    left = other.getLeft();
64    right = other.getRight();
65  }
66
67  OurCSCE310Tree::~OurCSCE310Tree(){
68    delete left;
69    left = NULL;
70    delete right;
71    right = NULL;
72    value = 0;
73  }
```

```
74
75 OurCSCE310Tree* OurCSCE310Tree::getParent(){
76   return parent;
77 }
78
79 OurCSCE310Tree* OurCSCE310Tree::getLeft(){
80   return left;
81 }
82
83 OurCSCE310Tree* OurCSCE310Tree::getRight(){
84   return right;
85 }
86
87 int OurCSCE310Tree::getValue(){
88   return value;
89 }
90
91 void OurCSCE310Tree::setParent( OurCSCE310Tree* par ){
92   parent = par;
93 }
94
95 void OurCSCE310Tree::setLeft( OurCSCE310Tree* lft ){
96   left = lft;
97 }
98
99 void OurCSCE310Tree::setRight( OurCSCE310Tree* rght ){
100  right = rght;
101 }
102
103 void OurCSCE310Tree::setValue( int val ){
104  value = val;
105 }
106
107 void OurCSCE310Tree::insert( int val ){
108  if( !getValue() ){
109    setValue( val );
110  }
111  else if( ( val < getValue() && !getLeft() ) || ( val < getValue() && !getLeft()->
      getValue() ) ){
112    left = new OurCSCE310Tree();
113    left->setParent( this );
114    left->setValue( val );
115  }
116  else if( ( val > getValue() && !getRight() ) || ( val > getValue() && !getRight()->
      getValue() ) ){
117    right = new OurCSCE310Tree();
118    right->setParent( this );
119    right->setValue( val );
120  }
121  else if( val < getValue() ){
122    getLeft()->insert( val );
123  }
124  else{
125    getRight()->insert( val );
126  }
127
128  if( getLeft() && getLeft()->getRight() && !getRight() || getLeft() && getLeft()->
      getRight() && getRight() && getLeft()->getHeight() > getRight()->getHeight() + 1
      && getLeft()->getRight()->getHeight() > getLeft()->getLeft()->getHeight() + 1 ){
129    rotateLeftRight();
```

```
130   }
131   else if( getRight() && getRight()->getLeft() && !getLeft() || getRight() && getRight
       ()->getLeft() && getLeft() && getRight()->getHeight() > getLeft()->getHeight() + 1
        && getRight()->getLeft()->getHeight() > getRight()->getRight()->getHeight() + 1 )
       {
132     rotateRightLeft();
133   }
134   else if( getLeft() && !getRight() && getLeft()->getHeight() > 1 || getLeft() &&
       getRight() && getLeft()->getHeight() > getRight()->getHeight() + 1 ){
135     rotateRight();
136   }
137   else if( getRight() && !getLeft() && getRight()->getHeight() > 1 || getRight() &&
       getLeft() && getRight()->getHeight() > getLeft()->getHeight() + 1 ){
138     rotateLeft();
139   }
140 }
141
142 void OurCSCE310Tree::printPreorder(){
143   if( getValue() ){
144     cout << getValue();
145   }
146   if( getLeft() && getLeft()->getValue() ){
147     cout << ",";
148     getLeft()->printPreorder();
149   }
150   if( getRight() && getRight()->getValue() ){
151     cout << ",";
152     getRight()->printPreorder();
153   }
154 }
155
156 void OurCSCE310Tree::printInorder(){
157   if( getLeft() && getLeft()->getValue() ){
158     getLeft()->printInorder();
159     cout << ",";
160   }
161   if( getValue() ){
162     cout << getValue();
163   }
164   if( getRight() && getRight()->getValue() ){
165     cout << ",";
166     getRight()->printInorder();
167   }
168 }
169
170 void OurCSCE310Tree::printPostorder(){
171   if( getLeft() && getLeft()->getValue() ){
172     getLeft()->printPostorder();
173     cout << ",";
174   }
175   if( getRight() && getRight()->getValue() ){
176     getRight()->printPostorder();
177     cout << ",";
178   }
179   if( getValue() ){
180     cout << getValue();
181   }
182 }
183
184 int OurCSCE310Tree::getHeight(){
```

```
185   if( getLeft() && getLeft()->getValue() && ( !getRight() || !getRight()->getValue() )
        ){
186     return getLeft()->getHeight() + 1;
187   }
188   else if( getRight() && getRight()->getValue() && ( !getLeft() || !getLeft()->
       getValue() ) ){
189     return getRight()->getHeight() + 1;
190   }
191   else if( getRight() && getLeft() && getRight()->getValue() && getLeft()->getValue()
       ){
192     return fmax( getRight()->getHeight() , getLeft()->getHeight() ) + 1;
193   }
194   else if( getValue() && ( !getLeft() || !getLeft()->getValue() ) && ( !getRight() ||
       !getRight()->getValue() ) ){
195     return 1;
196   }
197   return 0;
198 }
199
200 void OurCSCE310Tree::rotateLeftRight(){
201   getLeft()->rotateLeft();
202   rotateRight();
203 }
204
205 void OurCSCE310Tree::rotateRightLeft(){
206   getRight()->rotateRight();
207   rotateLeft();
208 }
209
210 void OurCSCE310Tree::rotateLeft(){
211
212   int tempRight = this->getRight()->getValue();
213   OurCSCE310Tree* rightNode = this->getRight();
214   OurCSCE310Tree* leftNode = this->getLeft();
215
216   this->getRight()->setValue(this->getValue());
217   this->setValue(tempRight);
218
219   this->setRight(this->getRight()->getRight());
220
221   this->setLeft(rightNode);
222   this->getLeft()->setRight(rightNode->getLeft());
223   this->getLeft()->setLeft(leftNode);
224
225 }
226
227 void OurCSCE310Tree::rotateRight(){
228
229   int tempLeft = this->getLeft()->getValue();
230   OurCSCE310Tree* leftNode = this->getLeft();
231   OurCSCE310Tree* rightNode = this->getRight();
232
233   this->getLeft()->setValue(this->getValue());
234   this->setValue(tempLeft);
235
236   this->setLeft(this->getLeft()->getLeft());
237
238   this->setRight(leftNode);
239   this->getRight()->setLeft(leftNode->getRight());
240   this->getRight()->setRight(rightNode);
```

```
241
242 }
243
244 void OurCSCE310Tree::deleteNode( int key ){
245
246   //delete root
247   OurCSCE310Tree* node = this;
248   if (node->getValue() == key) {
249     while (node->getLeft() != NULL) {
250       node = node->getLeft();
251     }
252     this->setValue(node->getValue());
253   }
254
255 }
```