COMP 2402 w22

Assignment 3 Workshop

Part A

Q1:

Example 1:

```
x = 3 → this tells us the length of the chain is 3
input:
      5 → first element in the chain is always index 0
          element % length = 5 % 6 = 5 → so the next element
      is
          at index 5
        → third and final element

      4

      8

      10

      2

      6 → second element in the chain
          element % length = 6 % 6 = 0 → so next element is
      at
          index 0

Chain:
      [5, 6, 5]

(5 + 6 + 5) mod 2402 = 16

output: 16
```

Example 2:

```
x = 3
input:
      5 → first element in the chain
          5 % 6 = 5 → got to index 5

      4 → third and final element

      8

      10

      2

      7 → second element in the chain
          7 % 6 = 1 → go to index 1

Chain:
      [5, 7, 4]

(5 + 7 + 4) mod 2402 = 16

output: 16
```

Example 3:

```
x = 0
Input:
  1
  2
  3
  4
  5

Output: 0 ← Since x = 0
```

Example 4:

```
x = 1
Input:
  1
  2
  3
  4
  5

Output: 1
```

Q2:

Example 1:

x = 3 → this tells us the length of the chain is 3

input:

5 → first element in the chain
5 % 6 = 5 → got to index 5
→ third and final element

4

8

10

2

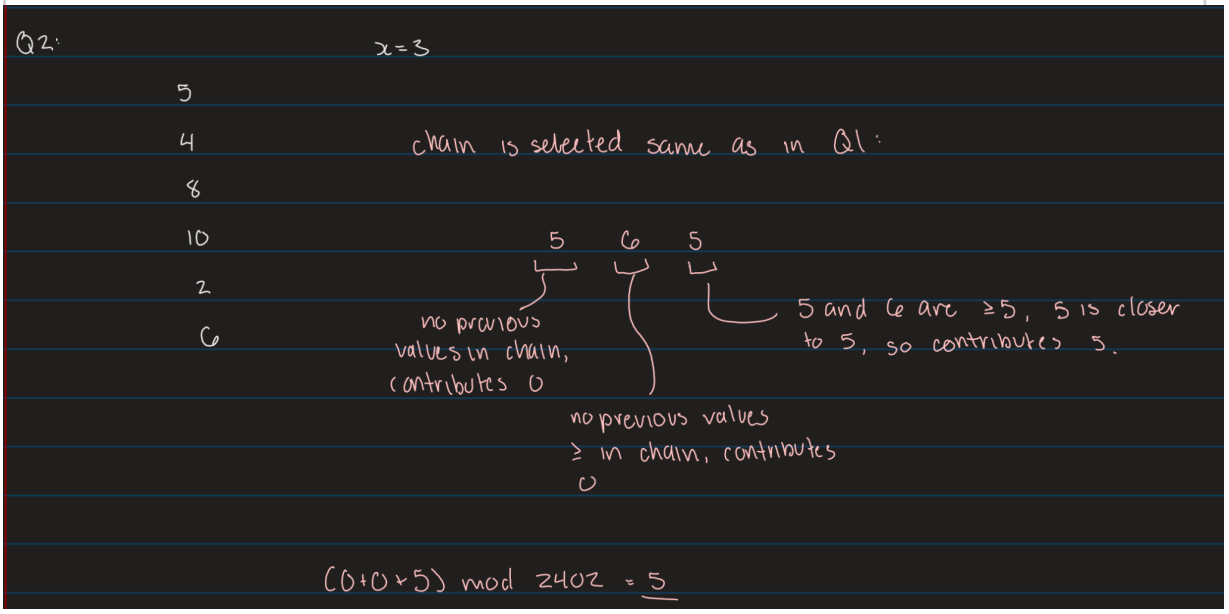
6 → second element in the chain
6 % 6 = 0 → go to index 0

Chain:
[5, 6, 5]

5 (index 0) contributes 0 (no previous values)
6 (index 1) contributes 0 (only previous value = 5 < 6)
5 (index 2) contributes 5 (5 >= 5 and closer to 5 than 6)

(0 + 0 + 5) mod 2402 = 5

output: 5



Example 2:

```
x = 3 → this tells us the length of the chain is 3
Input:
index
   0   5 → 1st element in the chain, 5%6=5 → go to index 5
   1   4 → Third element in chain
   2   8
   3   10
   4   2
   5   1 → second element in the chain
           1 % 6 = 1 → go to index 1

Chain:
      [5, 1, 4]

5 (index 0) contributes 0 (no previous values)
1 (index 1) contributes 5 (1 < 5)
4 (index 2) contributes 5 (4 < 5)


(0 + 5 + 5) mod 2402 = 10
```

output: 10

Example 3:

```
x = 0
Input:
     1
     2
     3

Output :  0 since x = 0
```

Example 4:

```
x = 1
Input:
     1
     2
     3

Output :  0 since x = 1  contributes 0 (no previous values)
```

Q3:

Example 1:

x = 3 → this tells us the length of the chain is 3 and we're inserting x-1=2 times

input:

8
4
7
10
2
9

Step 1: Get first element in the chain

8 → first element in the chain
Get value to insert:
l.size()-1-i = 6 - 1 - 0 = 5
value at index 5 = 9
Get position to insert at:
(i+1)%l.size() = (0+1)%6 = 1
Insert 9 at index 1

4
7
10
2
9

Step 2: Insert first time

8
9 → inserted 9 at index 1
4
7
10
2
9

Step 3: Get second element in chain

element % length = 8 % 7 = 1 → go to position 1 for 2nd element

8
9 → second element in the chain
Get value to insert:
l.size()-1-i = 7 - 1 - 1 = 5
value at index 5 = 2
Get position to insert at:

```
            (i+1)%l.size() = (1+1)%7 = 2
        Insert 2 at index 2
    4
    7
    10
    2
    9
```

Step 4: Insert second time

```
    8
    9
    2 → inserted 2 at index 2
    4
    7
    10
    2
    9
```

Step 5: Get third (final) element in chain

```
element % length = 9 % 8 = 1 → go to position 1 for 3nd element
    8
    9 → third element in the chain
        Note we don't need to get value to insert this time
        because we have completed the chain, any other work
        on this list is now unnecessary
    2
    4
    7
    10
    2
    9
```

```
Chain:
    [8, 9, 9]

(8 + 9 + 9) mod 2402 = 26
```

output: **26**

Q4:

Example 1:

```
      x = 5
First step:
           index
      10     0    ← 1st element of chain 10 mult of 5(correct)
      20     1
      5      2
      3      3
      2      4
      24     5

Chain: 10

Second step:

           index
      10     0    ← (10 % 6) = 4 next element is index 4
      20     1
      5      2
      3      3
      2      4     ← 2nd element, check the chain, add 10 to sum, add
2 to chain
      24     5

Chain: 10, 2
Sum: 10

Third step:

           index
      10     0
      20     1
      5      2 ← check chain, add 10 to sum, add 5 to chain then 5
mult 5 is correct
      3      3
      2      4     ← 2 % 6 = 2 , next element is index 2
      24     5

Chain: 10, 2, 5
Sum: 10, 10

Fourth step:

           index
```

```
     10      0
     20      1
     5       2 ← 5 % 6 = 5   , next element is index 5
     3       3
     2       4
     24      5 ← check chain,  add 5 to sum, and 24 div by 24,
remove the minimum multiple of 5 in chain, add 24 to chain

Sum: 10, 10, 5
Chain: 10, 2, 5← remove , 24
Chain:10, 2, 24

Fifth step:

          index
     10      0   ← check chain, add 10 to sum, and add 10 to chain
     20      1
     5       2
     3       3
     2       4
     24      5 ← 24 % 6 = 0 ← next element is index 0

Sum: 10, 10, 5,10
Chain: 10, 2, 24
Chain:10, 2, 24,10
```

Part B

Q5:

Example 1:

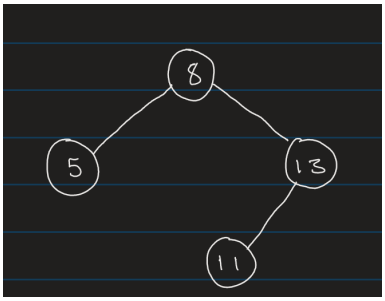
```
this = {4, 6, 8, 10}
other = {1, 2, 5}

{4+1, 6+2, 8+5, 10+1} = {5, 8, 13, 11} → {5, 8, 11, 13}
```

Note that this is a set and not a list:

- No duplicates
- The order in which we write it is not important

Hint: be sure to thoroughly read through the BinarySearchTree class → there is an iterator method

Possible visual representation of output:



Note it could look different → remember that a binary search tree is made up of nodes which have a left and right child whose values were not established above, so this tree might look a bit different.

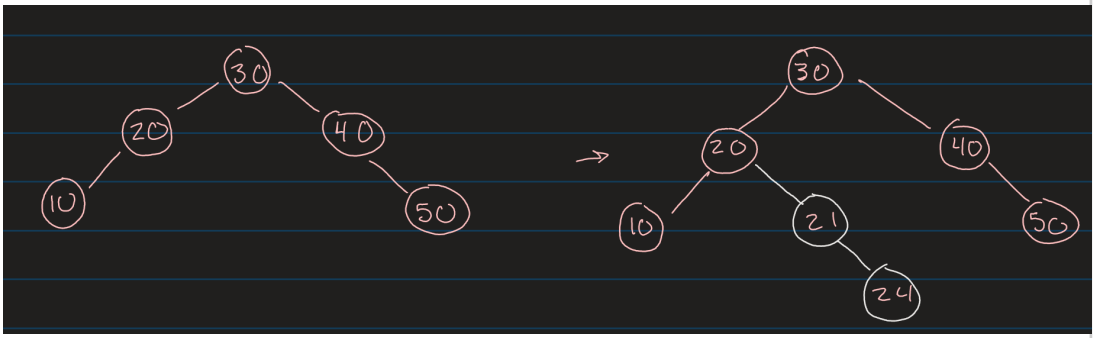
This problem is similar to assignment 1 question 6 and assignment 2 question 5. You can see more “edge cases” and examples from the previous workshops.

The difference / challenge with sum() using binary search trees, is that you’re working with sets now as opposed to lists and each element in the set is actually a node - be sure to read the node class.

Q6:

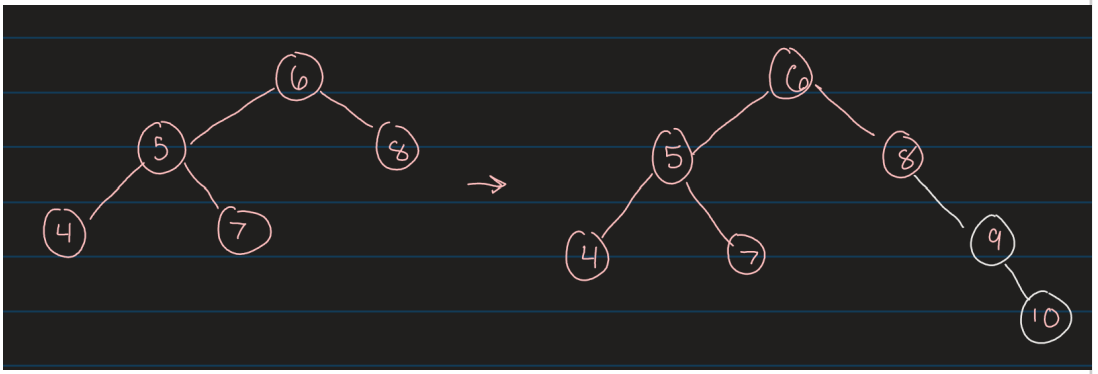
Example 1:

```
this = {10, 20, 30, 40, 50}
other = {21, 24}

{10, 20, 21, 24, 30, 40, 50}

Possible visual representation of output:
```



Example 2:

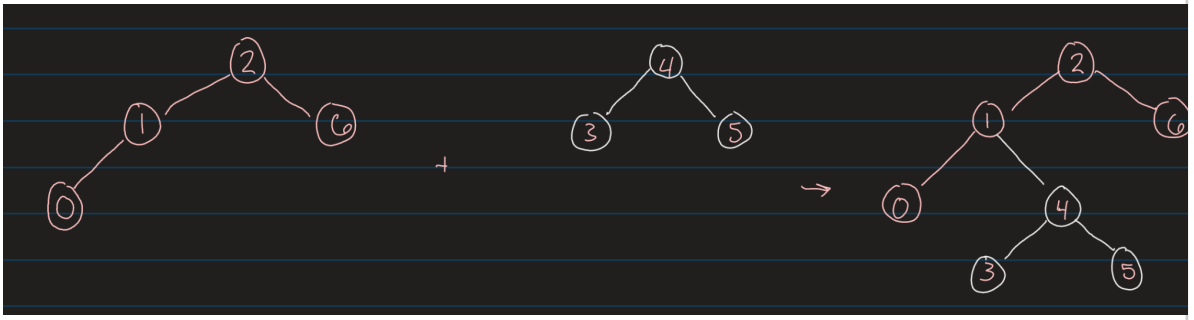
```
this = {4, 5, 6, 7, 8}
other = {9, 10}

{4, 5, 6, 7, 8, 9, 10}

Possible visual representation of output:
```



Example 3:

```
this = {0, 1, 2, 6}
```

```
other = {3, 4, 5}

{0, 1, 2, 3, 4, 5, 6}

Possible visual representation of output:
```



Example 4:

```
this = {0, 1, 2, 6}
other = {}

{0, 1, 2, 6}
```

Example 5:

```
this = {}
other = {1, 2}

{1, 2}
```

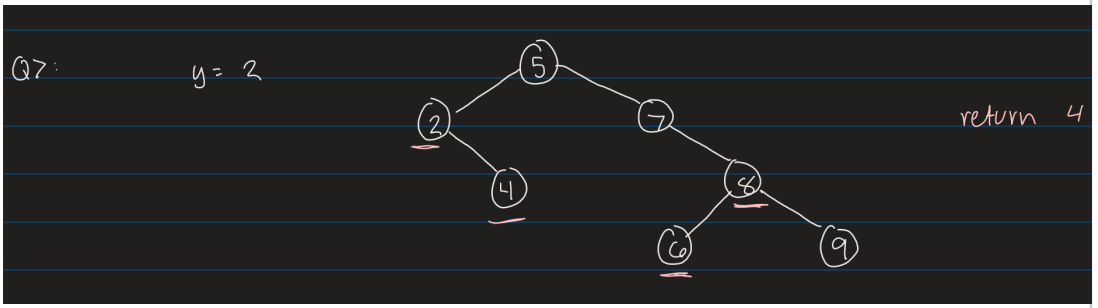
This question is similar to assignment 2 questions 7 and 9, but with binary search trees now.

Q7:

Example 1:

```
this = {2, 4, 5, 6, 7, 8, 9}
y = 2

2 % 2 = 0
4 % 2 = 0
6 % 2 = 0
8 % 2 = 0

output: 4
```

Possible visual representation of output:



This problem is hopefully pretty straightforward to understand, the challenge will be implementing a time efficient algorithm.

Note: you can't use recursion for this problem.

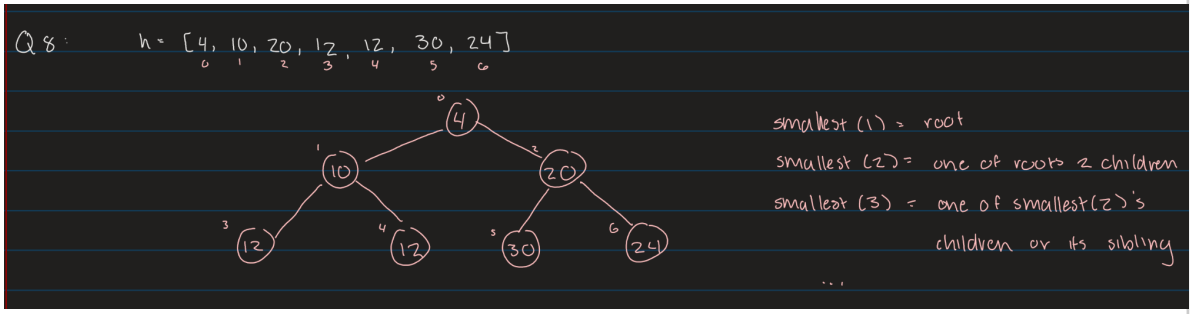
Example 2:

```
this = {3, 5, 7}
y = 2

output: 0
```

Q8:

Example 1:

```
this = [4, 10, 20, 12, 12, 30, 24]

smallest(1) = root = 4
smallest(2) = 10
smallest(3) = 12
smallest(4) = 12
smallest(5) = 20
smallest(6) = 24
smallest(7) = 30

Possible visual representation of output:
```



Note: you can assume k is an integer between 1 and size