COIS 3020 - Data Structures & Algorithms III

Assignment 3: Ternary Trees & Lazy Binomial Heaps

8th April 2022

Group Members:
Binh Nguyen (0710969)
Sruthi Sugumaran (0690271)
Marc Touma (0674382)
Brandon Van Baah (0658848)

Table of Contents

Part 1 - Ternary Tree Implementation

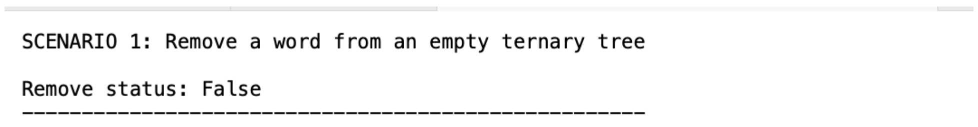
Note: Remove test cases include the print output of the printTree method as well.

Test Case 1.1 - “Remove”

Scenario 1: Remove a word from an empty ternary tree	

Input	Trie<int> T = new Trie<int>();
	Console.WriteLine($"Remove status: {T.Remove("Hi")}");

Expected Output	Remove status: False

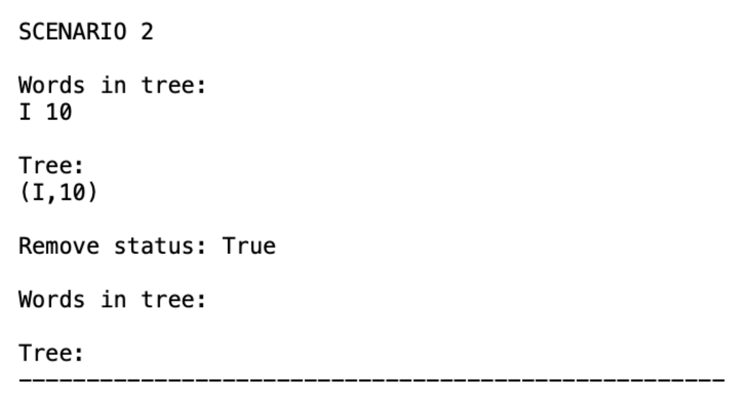
Actual Output	
	`SCENARIO 1: Remove a word from an empty ternary tree`
	
	`Remove status: False`
	`---------------------------------------------------`

Status	Success

Scenario 2: Remove a word from the tree when size = 1;	

Input	T.Insert("I", 10);
	Console.WriteLine($"\nRemove status: {T.Remove("I")}");

Expected Output	Remove status: True

Actual Output	
	`SCENARIO 2`
	
	`Words in tree:`
	`I 10`
	
	`Tree:`
	`(I,10)`
	
	`Remove status: True`
	
	`Words in tree:`
	
	`Tree:`
	`----------------------------------------------------`

Status	Success

Scenario 3: Removing a word from the tree when size > 1	
Input	Console.WriteLine($"\nRemove status: {T.Remove("**beet**")}");
Expected Output	Remove status: True
Actual Output	

```
SCENARIO 3

Words in tree:
abc 60
bad 90
bag 10
bagel 30
bat 20
beet 40
cab 70

Tree:
        (c, 0)  (a, 0)  (b,70)

                (e, 0)  (e, 0)  (t,40)

                        (t,20)

(b, 0)  (a, 0)  (g,10)  (e, 0)  (l,30)

                        (d,90)

        (a, 0)  (b, 0)  (c,60)


Remove status: True

Words in tree:
abc 60
bad 90
bag 10
bagel 30
bat 20
cab 70

Tree:
        (c, 0)  (a, 0)  (b,70)

                        (t,20)

(b, 0)  (a, 0)  (g,10)  (e, 0)  (l,30)

                        (d,90)

        (a, 0)  (b, 0)  (c,60)

----------------------------------------------------
```

Status	Success

Scenario 4: Removing a word whose last character is not a leaf

Input	Console.WriteLine($"\nRemove status: {T.Remove("**bag**")}");
Expected Output	Remove status: True

Actual Output

```
SCENARIO 4

Words in tree:
abc 60
bad 90
bag 10
bagel 30
bat 20
cab 70

Tree:
        (c, 0)  (a, 0)  (b,70)

                        (t,20)

(b, 0)  (a, 0)  (g,10)  (e, 0)  (l,30)

                        (d,90)

        (a, 0)  (b, 0)  (c,60)


Remove status: True

Words in tree:
abc 60
bad 90
bagel 30
bat 20
cab 70

Tree:
        (c, 0)  (a, 0)  (b,70)

                        (t,20)

(b, 0)  (a, 0)  (g, 0)  (e, 0)  (l,30)

                        (d,90)

        (a, 0)  (b, 0)  (c,60)
```

Status	Success

Scenario 5: Removing a word whose characters have middle, low and high nodes	
Input	Console.WriteLine($"\nRemove status: {T.Remove("**bagel**")}");
Expected Output	Remove status: True

Actual Output	
	```
	SCENARIO 5

	Words in tree:
	abc 60
	bad 90
	bagel 30
	bat 20
	cab 70

	Tree:
	        (c, 0)  (a, 0)  (b,70)

	                        (t,20)

	(b, 0)  (a, 0)  (g, 0)  (e, 0)  (l,30)

	                        (d,90)

	        (a, 0)  (b, 0)  (c,60)


	Remove status: True

	Words in tree:
	abc 60
	bad 90
	bat 20
	cab 70

	Tree:
	        (c, 0)  (a, 0)  (b,70)

	                        (t,20)

	(b, 0)  (a, 0)  (d,90)

	        (a, 0)  (b, 0)  (c,60)

	----------------------------------------------------
	```

Status	Success

Scenario 6: Removing a word whose characters does not have low and high nodes

Input	Console.WriteLine($"\nRemove status: {T.Remove("**abc**")}");
Expected Output	Remove status: True

Actual Output

```
SCENARIO 6

Words in tree:
abc 60
bad 90
bat 20
cab 70

Tree:
        (c, 0)  (a, 0)  (b,70)

                        (t,20)

(b, 0)  (a, 0)  (d,90)

        (a, 0)  (b, 0)  (c,60)


Remove status: True

Words in tree:
bad 90
bat 20
cab 70

Tree:
        (c, 0)  (a, 0)  (b,70)

                        (t,20)

(b, 0)  (a, 0)  (d,90)

----------------------------------------------------
```

Status	Success

Scenario 7: Removing a word that does not exist in the ternary tree

Input	Console.WriteLine($"\nRemove status: {T.Remove("**bagel**")}");
Expected Output	Remove status: False

Actual Output

```
SCENARIO 7

Words in tree:
bad 90
bat 20
cab 70

Tree:
        (c, 0)  (a, 0)  (b,70)

                        (t,20)

(b, 0)  (a, 0)  (d,90)


Remove status: False

Words in tree:
bad 90
bat 20
cab 70

Tree:
        (c, 0)  (a, 0)  (b,70)

                        (t,20)

(b, 0)  (a, 0)  (d,90)

----------------------------------------------------
```

Status	Success

Part 2 - Lazy Binomial Heaps

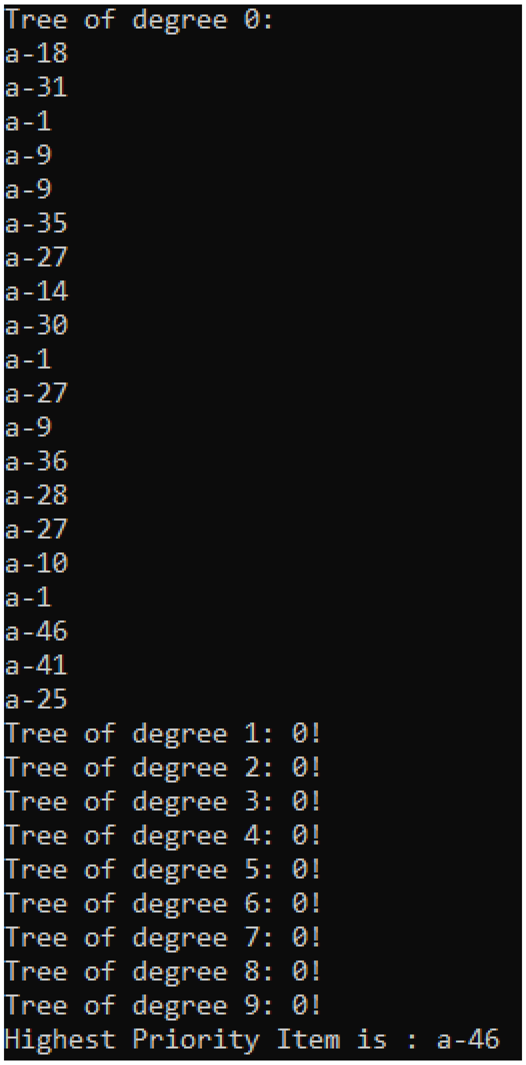
Console.WriteLine("Tree of degree {0}: 0!",i);
[Test Case 1] or “”
Description:

Scenario1: Try removing from an empty heap	
Input	BH.Remove()
Expected Output	error
Actual Output	`error, heap is empty`
Status	True

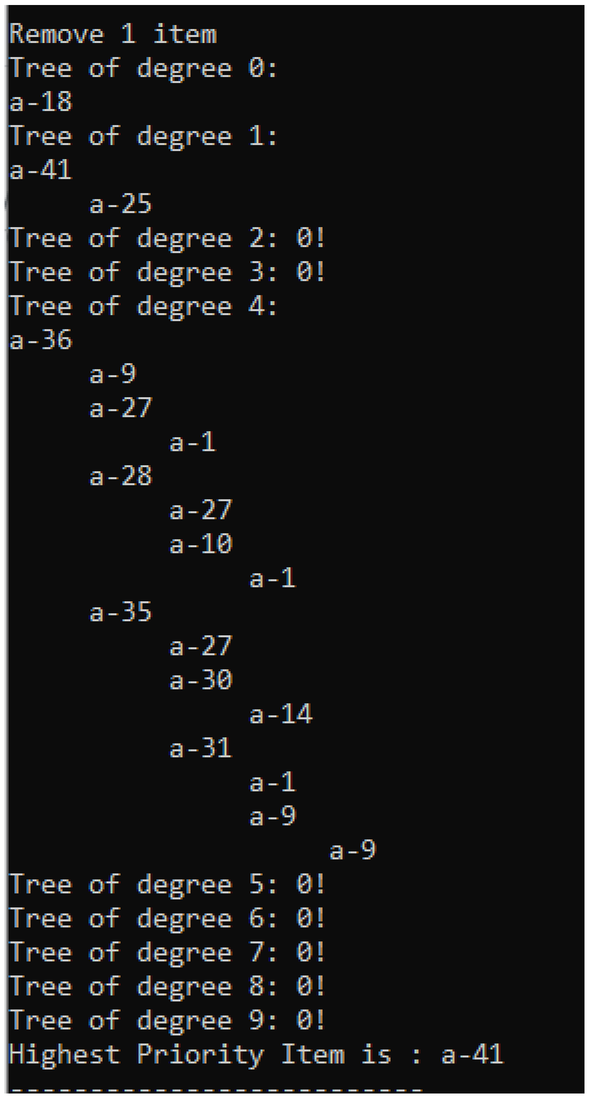
Scenario 2: create a heap, and insert 20 numbers	

Input	`for (i = 0; i < 20; i++)`
	`{`
	`BH.Add(new PriorityClass(r.Next(50), (char)('a')));`
	`}`

Expected Output	the first array contains the Binomial Trees of degree 0 of all the inserted numbers

Actual Output	 `Tree of degree 0:` `a-18` `a-31` `a-1` `a-9` `a-9` `a-35` `a-27` `a-14` `a-30` `a-1` `a-27` `a-9` `a-36` `a-28` `a-27` `a-10` `a-1` `a-46` `a-41` `a-25` `Tree of degree 1: 0!` `Tree of degree 2: 0!` `Tree of degree 3: 0!` `Tree of degree 4: 0!` `Tree of degree 5: 0!` `Tree of degree 6: 0!` `Tree of degree 7: 0!` `Tree of degree 8: 0!` `Tree of degree 9: 0!` `Highest Priority Item is : a-46`
Status	True

Scenario 3: Remove 1 item from the heap and check that front changes	
Input	BH.Remove();
Expected Output	The heap has been coalesced, 46 has been removed, and front has been updated

Actual Output	

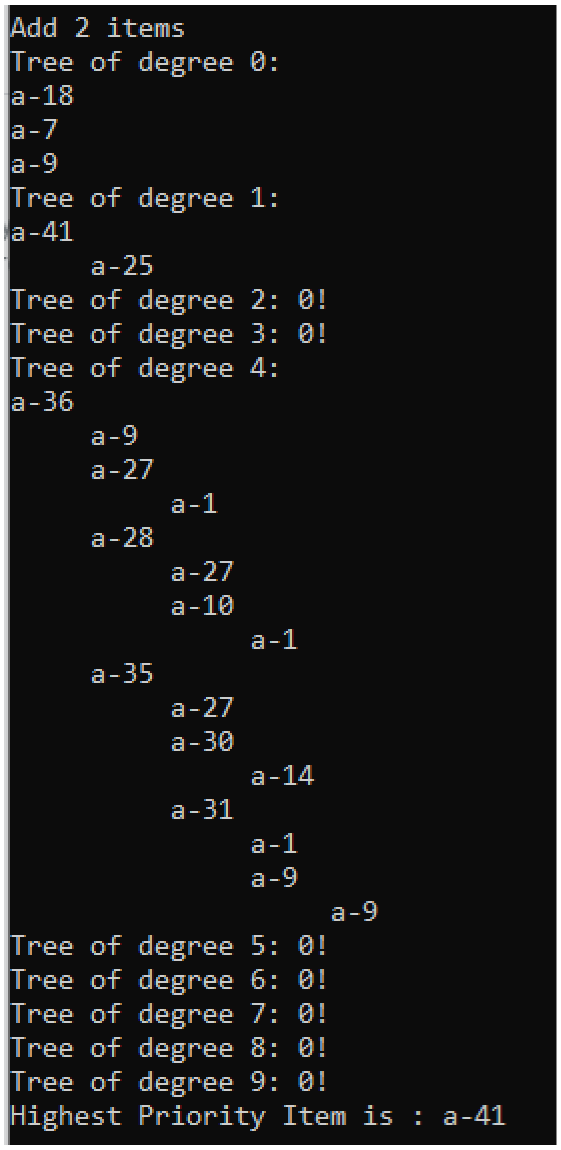
```
Remove 1 item
Tree of degree 0:
a-18
Tree of degree 1:
a-41
     a-25
Tree of degree 2: 0!
Tree of degree 3: 0!
Tree of degree 4:
a-36
     a-9
     a-27
          a-1
     a-28
          a-27
          a-10
               a-1
     a-35
          a-27
          a-30
               a-14
          a-31
               a-1
               a-9
                    a-9
Tree of degree 5: 0!
Tree of degree 6: 0!
Tree of degree 7: 0!
Tree of degree 8: 0!
Tree of degree 9: 0!
Highest Priority Item is : a-41
--------------------------
```

Status	True

Scenario 4: Add 2 items. Print out the heap. Test Front()	

Input	BH.Add(new PriorityClass(7, (char)('a')));
	BH.Add(new PriorityClass(9, (char)('a')));
	BH.Print();

Expected Output	the first array contains the Binomial Trees of degree 0 of all the inserted numbers

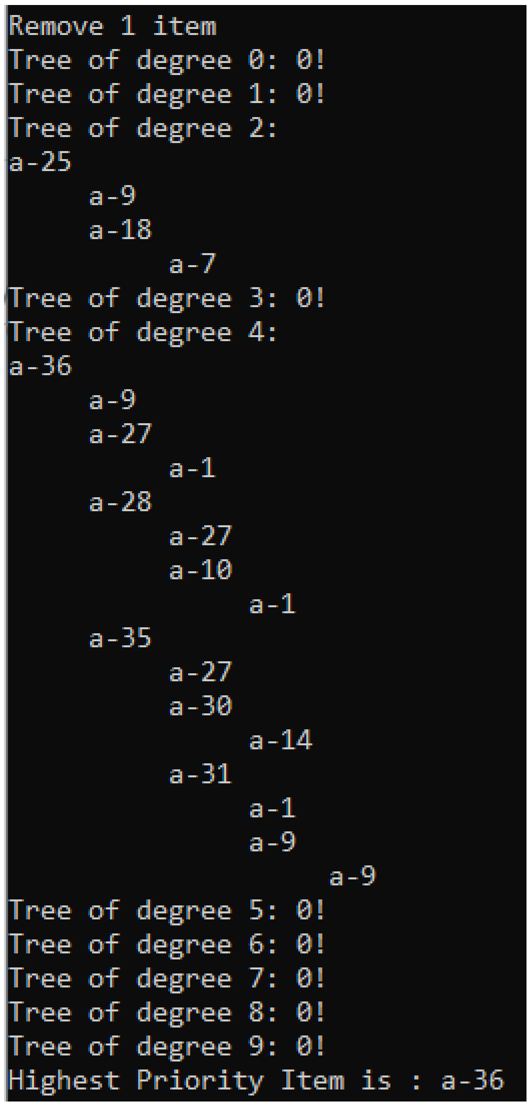
Actual Output	
	```
	Add 2 items
	Tree of degree 0:
	a-18
	a-7
	a-9
	Tree of degree 1:
	a-41
	     a-25
	Tree of degree 2: 0!
	Tree of degree 3: 0!
	Tree of degree 4:
	a-36
	     a-9
	     a-27
	          a-1
	     a-28
	          a-27
	          a-10
	               a-1
	     a-35
	          a-27
	          a-30
	               a-14
	          a-31
	               a-1
	               a-9
	                    a-9
	Tree of degree 5: 0!
	Tree of degree 6: 0!
	Tree of degree 7: 0!
	Tree of degree 8: 0!
	Tree of degree 9: 0!
	Highest Priority Item is : a-41
	```

Status	

Scenario 5: Remove 1 item. Print out heap. Test Front()	

Input	BH.Remove();
	BH.Print();

Expected Output	The heap has been coalesced, the node containing 41 has been removed, and Front has been updated

Actual Output	

```
Remove 1 item
Tree of degree 0: 0!
Tree of degree 1: 0!
Tree of degree 2:
a-25
     a-9
     a-18
          a-7
Tree of degree 3: 0!
Tree of degree 4:
a-36
     a-9
     a-27
          a-1
     a-28
          a-27
          a-10
               a-1
     a-35
          a-27
          a-30
               a-14
          a-31
               a-1
               a-9
                    a-9
Tree of degree 5: 0!
Tree of degree 6: 0!
Tree of degree 7: 0!
Tree of degree 8: 0!
Tree of degree 9: 0!
Highest Priority Item is : a-36
```

Status	True