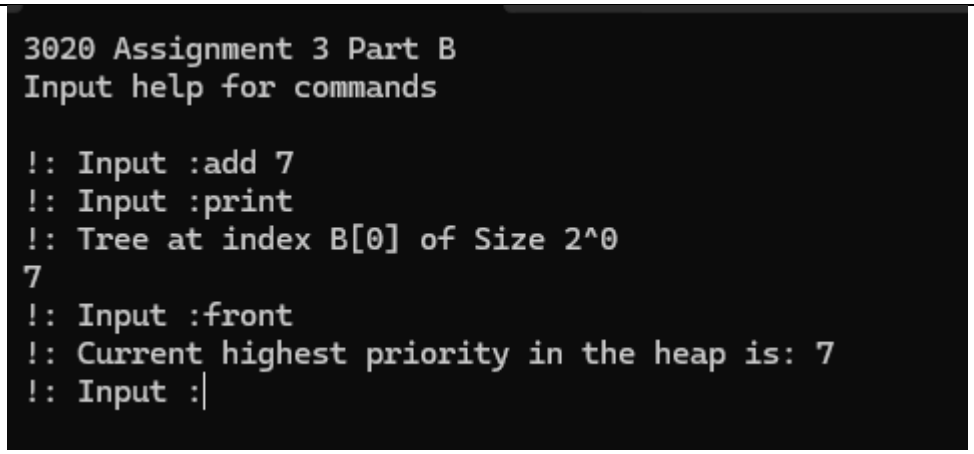
Assignment 3 Part B Testing Documentation
Test 1
Test Case: Testing basic add, print, and front methods
Input: add 7, print, front
Expected Output: 7 should be the only node in B[0] and the front should be 7 as well
Actual: 
Test 2
Test Case: Testing basic remove case by adding one item to the heap then removing
Input: add 7, print, remove, print, front
Expected Output: 7 should be added to the heap, print and front should work as they did in the previous example, when 7 is removed, print should say that the heap is empty, and when front is called the program should throw an exception

```
3020 Assignment 3 Part B
Input help for commands

!: Input :add 7
!: Input :print
!: Tree at index B[0] of Size 2^0
7
!: Input :front
!: Current highest priority in the heap is: 7
!: Input :remove
!: Input :print
!: The lazy heap is currently empty
!: Input :front
```

Actual:

COIS 3020 Assignment 3 Part B Testing Documentation
Matthew Hellard and Benjamin Macintosh

Test 3
Test Case: adding 4 nodes, then removing the highest
Input: add 1, add 2, add 3, add 4, print, front, remove, print, front
Expected Output: After adding the four nodes they should de displayed 4-3-2-1 (because the most recent as added to the front), and four should be the highest priority. After calling removed, 1 should be in index B[0] by itself, 3 should be in index B[1] with a child of 2. Then 3 should be the highest priority item in the heap.

Actual:

```
3020 Assignment 3 Part B
Input help for commands

!: Input :add 1
!: Input :add 2
!: Input :add 3
!: Input :add 4
!: Input :print
!: Tree at index B[0] of Size 2^0
4
4 Has Sibling: 3
3 Has Sibling: 2
2 Has Sibling: 1
!: Input :front
!: Current highest priority in the heap is: 4
!: Input :remove
!: Input :print
!: Tree at index B[0] of Size 2^0
1
!: Tree at index B[1] of Size 2^1
3
3 Has Child: 2
!: Input :front
!: Current highest priority in the heap is: 3
!: Input :
```

Test 4

Test Case: adding 5 nodes and then removing the item with the highest priority

Input: add 30, 26, 45, 60, 99, print, front , remove, print front

Expected Output: Before calling remove, the 5 items should all be siblings at B[0] with 99 being the highest priority. After remove. Since there are 4 items, they would each be put into trees of 2^1, which would be stored at B[1]. Then since there are two trees at B[1], they would be combined into one tree and stored at B[2]. And 60 would be the highest priority item in the heap

Actual:

```
3020 Assignment 3 Part B
Input help for commands

!: Input :add 30
!: Input :add 26
!: Input :add 45
!: Input :add 60
!: Input :add 99
!: Input :print
!: Tree at index B[0] of Size 2^0
99 - 60 - 45 - 26 - 30
!: Input :front
!: Current highest priority in the heap is: 99
!: Input :remove
!: Input :print
!: Tree at index B[2] of Size 2^2
60
|
30 - 45
|
26
!: Input :front
!: Current highest priority in the heap is: 60
!: Input :
```

Test 5

Test Case: Testing remove using the tree from the previous case, for one subtrees need to be broken up.

Input: remove, print, front

Expected Output: 45 should be by itself in B[0] and 30 should be in B[1] with 26 as its sole child. 45 should be the highest priority item in the heap.

Actual:

```
3020 Assignment 3 Part B
Input help for commands

!: Input :add 30
!: Input :add 26
!: Input :add 45
!: Input :add 60
!: Input :add 99
!: Input :print
!: Tree at index B[0] of Size 2^0
99
99 Has Sibling: 60
60 Has Sibling: 45
45 Has Sibling: 26
26 Has Sibling: 30
!: Input :front
!: Current highest priority in the heap is: 99
!: Input :remove
!: Input :print
!: Tree at index B[2] of Size 2^2
60
60 Has Child: 30
30 Has Sibling: 45
30 Has Child: 26
!: Input :front
!: Current highest priority in the heap is: 60
!: Input :
```

Test 6

Test Case: Testing with 20 nodes to ensure the lazy heap can handle larger data sets

Input: a 1, a 2, a, 3, a 4,… a 21, print, front, remove, print, front

Expected Output: Should have a tree of size 4 rooted in 4 and size 16 rooted at 20 stored in the appropriate indices. 4 at B[2] and 16 at B[4].

Actual:

```
3020 Assignment 3 Part B
Input help for commands

!: Input :a 1
!: Input :a 2
!: Input :a 3
!: Input :a 4
!: Input :a 5
!: Input :a 6
!: Input :a 7
!: Input :a 8
!: Input :a 9
!: Input :a 10
!: Input :a 11
!: Input :a 12
!: Input :a 13
!: Input :a 14
!: Input :a 15
!: Input :a 16
!: Input :a 17
!: Input :a 18
!: Input :a 19
!: Input :a 20
!: Input :a 21
!: Input :print
!: Tree at index B[0] of Size 2^0
21
21 Has Sibling: 20
20 Has Sibling: 19
19 Has Sibling: 18
18 Has Sibling: 17
17 Has Sibling: 16
16 Has Sibling: 15
15 Has Sibling: 14
14 Has Sibling: 13
13 Has Sibling: 12
12 Has Sibling: 11
11 Has Sibling: 10
10 Has Sibling: 9
9 Has Sibling: 8
8 Has Sibling: 7
7 Has Sibling: 6
6 Has Sibling: 5
5 Has Sibling: 4
```

```
4 Has Sibling: 3
3 Has Sibling: 2
2 Has Sibling: 1
!: Input :front
!: Current highest priority in the heap is: 21
!: Input :remove
!: Input :print
!: Tree at index B[2] of Size 2^2
4
4 Has Child: 2
2 Has Sibling: 3
2 Has Child: 1
!: Tree at index B[4] of Size 2^4
20
20 Has Child: 12
```