COIS 2020 Assignment 2 Part A Testing Document

Project: COIS 2020 — Assignment 2 Part A

Class under test: PriorityQueue (Binary heap)

Testers: Martin Wang (0843637) and Henry Smith (0850896)

Date: November 3rd, 2025

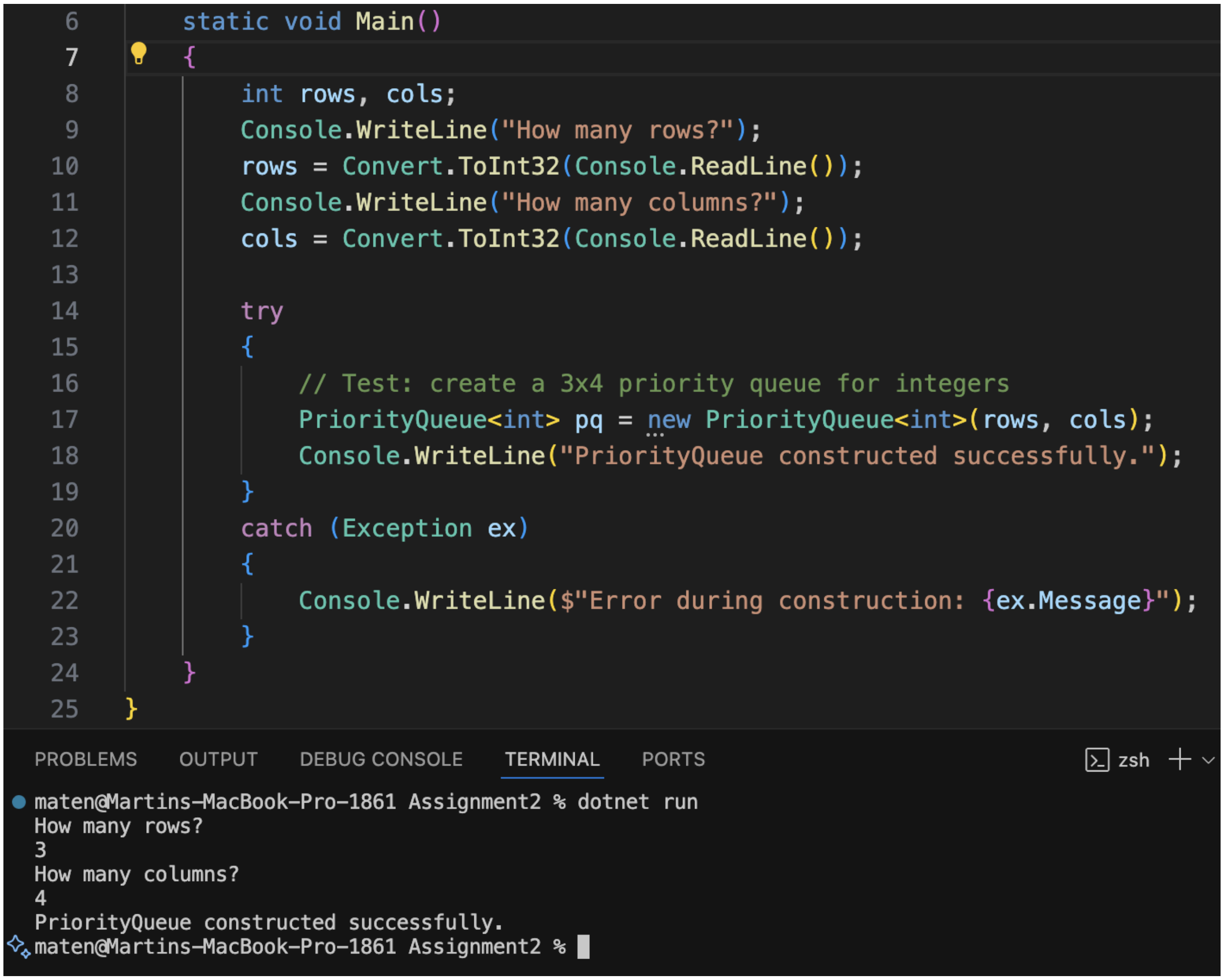
Approach: Implement and verify methods one by one in this order:

1. PriorityQueue(int m, int n) → 2) Print() → 3) Insert() → 4) Front() → 5) Remove() → 6) Found() → 7) Size()

Step 1 - PriorityQueue(int m, int n)

Test 1 - construct 3x4 heap matrix

- **Description**: Initialize a heap matrix to ensure the constructor works
- **Input**: “3”, “4”
- **Expected outcome**: PriorityQueue constructed successfully
- **Actual outcome**: PriorityQueue constructed successfully ✅

- 

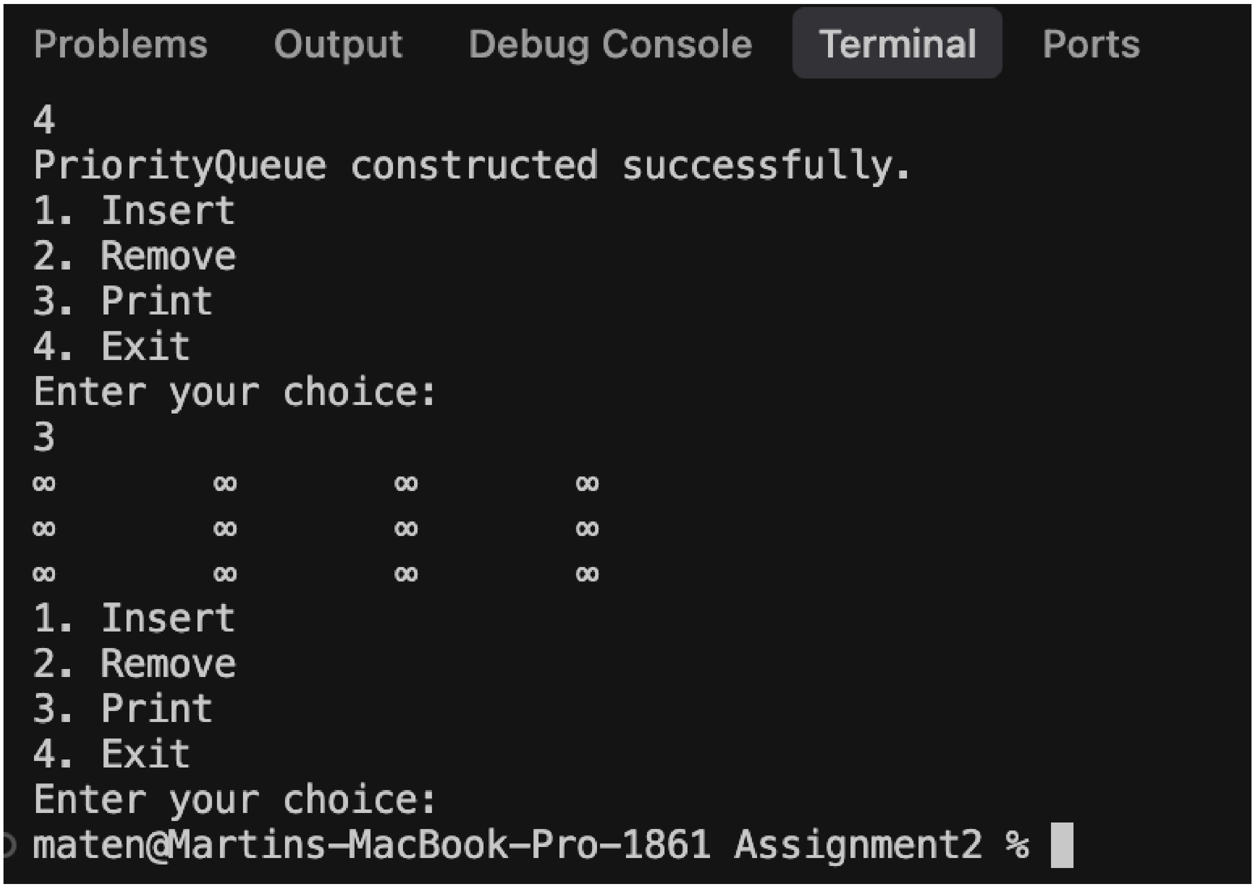
```csharp
    static void Main()
    {
        int rows, cols;
        Console.WriteLine("How many rows?");
        rows = Convert.ToInt32(Console.ReadLine());
        Console.WriteLine("How many columns?");
        cols = Convert.ToInt32(Console.ReadLine());

        try
        {
            // Test: create a 3x4 priority queue for integers
            PriorityQueue<int> pq = new PriorityQueue<int>(rows, cols);
            Console.WriteLine("PriorityQueue constructed successfully.");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Error during construction: {ex.Message}");
        }
    }
}
```

```
maten@Martins-MacBook-Pro-1861 Assignment2 % dotnet run
How many rows?
3
How many columns?
4
PriorityQueue constructed successfully.
maten@Martins-MacBook-Pro-1861 Assignment2 %
```

Step 2 - Print()

Test 1 - Print 3x4 matrix

- **Description:** Output the matrix
- **Input:** 3
- **Expected outcome:** output 3x4 matrix with infinity symbols as null
- **Actual outcome:** ✅

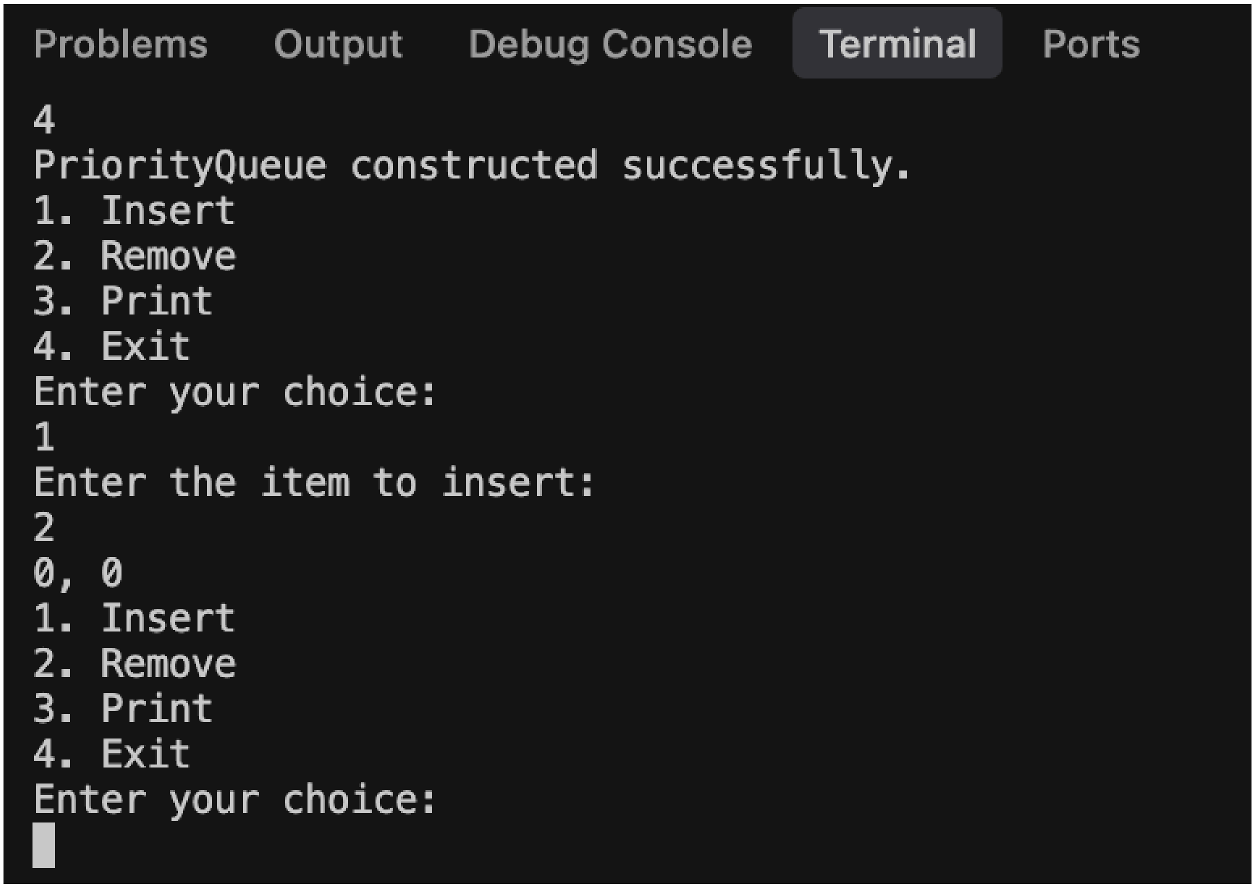


```
4
PriorityQueue constructed successfully.
1. Insert
2. Remove
3. Print
4. Exit
Enter your choice:
3
∞       ∞       ∞       ∞
∞       ∞       ∞       ∞
∞       ∞       ∞       ∞
1. Insert
2. Remove
3. Print
4. Exit
Enter your choice:
maten@Martins-MacBook-Pro-1861 Assignment2 %
```

Step 3 - Insert()

Test 1.1 - Convert index number to position coordinates

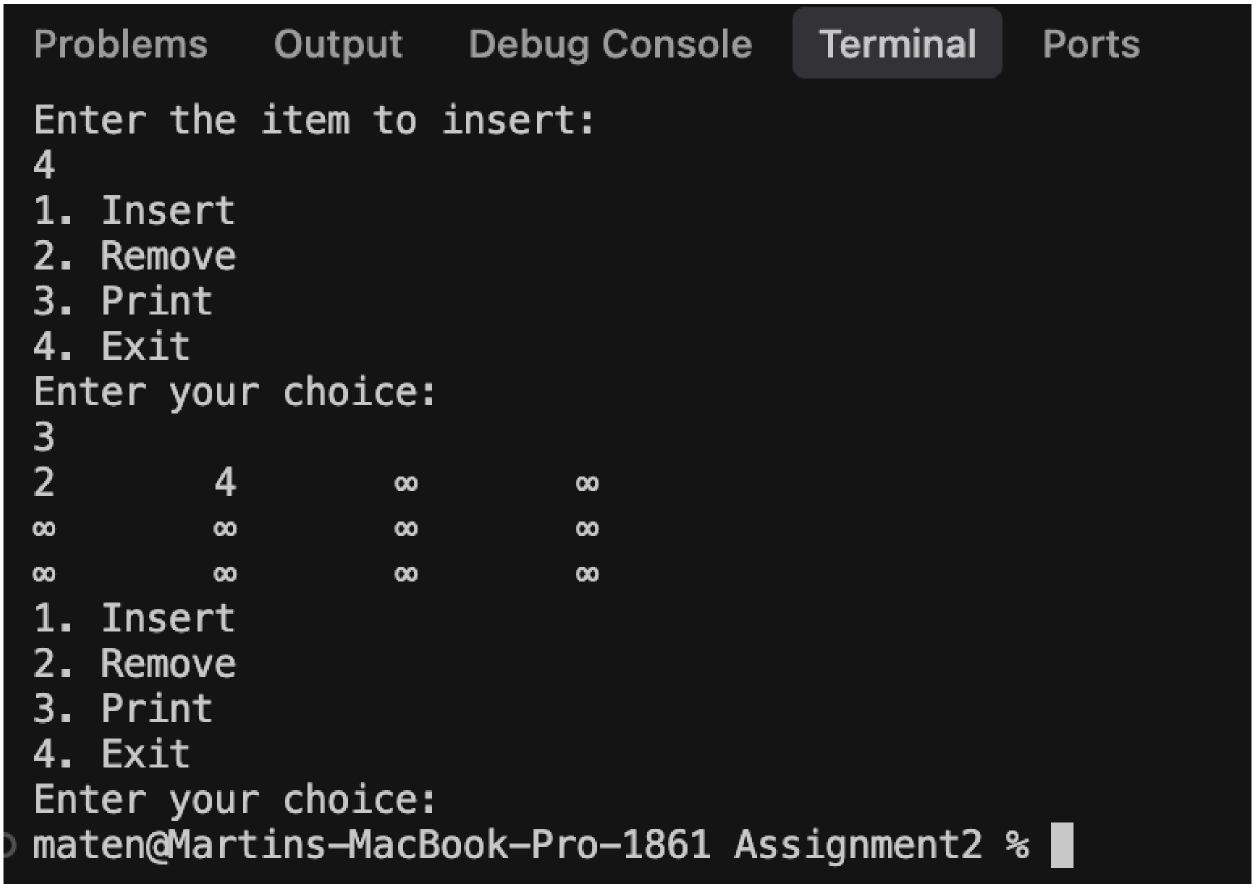
- **Description:** Using the helper function to determine matrix coordinates using index number and size of the matrix
- **Input:** 1, 2
- **Expected outcome:** 0, 0
- **Actual outcome:** ✅



```
4
PriorityQueue constructed successfully.
1. Insert
2. Remove
3. Print
4. Exit
Enter your choice:
1
Enter the item to insert:
2
0, 0
1. Insert
2. Remove
3. Print
4. Exit
Enter your choice:
```

Test 1.2 - Insert item into coordinate

- **Description:** replace the element at the coordinate position with the entered number
- **Input:** 1, 2, 1, 4, 3
- **Expected Outcome:** 3x4 Matrix printed with the first element replaced by 2, second element replaced by 4
- **Actual outcome:** ✅

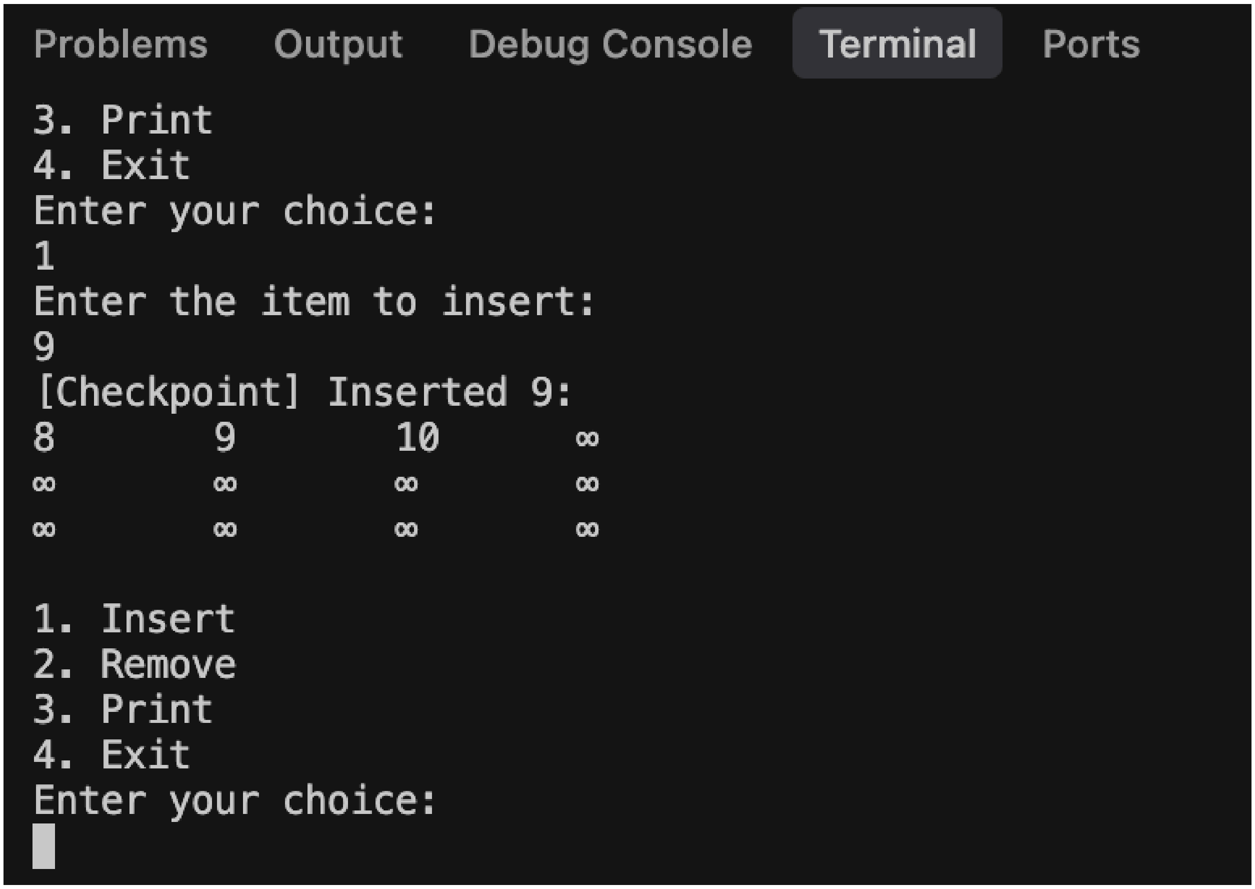


```
Enter the item to insert:
4
1. Insert
2. Remove
3. Print
4. Exit
Enter your choice:
3
2       4       ∞       ∞
∞       ∞       ∞       ∞
∞       ∞       ∞       ∞
1. Insert
2. Remove
3. Print
4. Exit
Enter your choice:
maten@Martins-MacBook-Pro-1861 Assignment2 %
```

Test 1.3 - Sort the numbers

- **Description:** Use a helper function (BubbleUp) to sort the priority of the numbers. The lowest number being on the top left, increasing as the priority lowers.
- **Input:** 1, 10, 1, 8, 1, 9
- **Expected Outcome:** 3x4 Matrix printed with the numbers sorted in the order of 8, 9, 10

- **Actual Outcome:** ✅

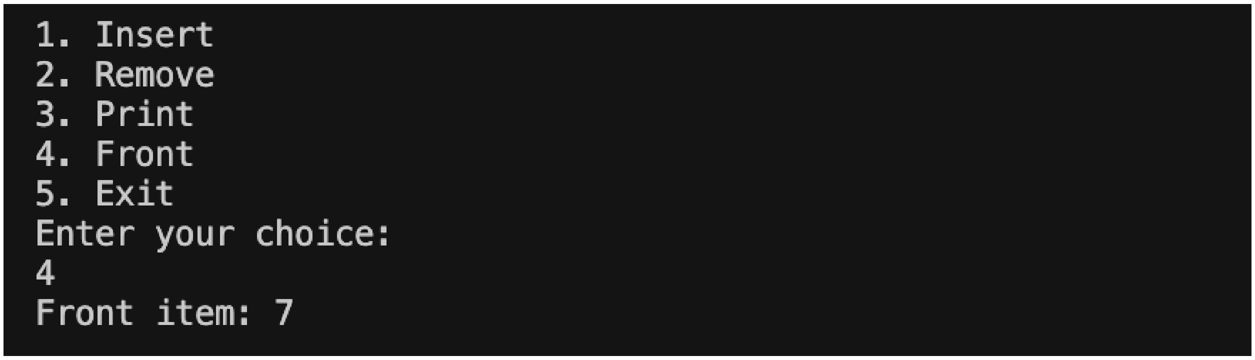


```
3. Print
4. Exit
Enter your choice:
1
Enter the item to insert:
9
[Checkpoint] Inserted 9:
8       9       10      ∞
∞       ∞       ∞       ∞
∞       ∞       ∞       ∞

1. Insert
2. Remove
3. Print
4. Exit
Enter your choice:
█
```

Step 4 - front()

Test 1.1 - Retrieve the value with the highest priority

- **Description:** Get the value at position 0,0 since this would be the value with the highest priority
- **Input:** (pre-existing 3x4 matrix made up of numbers 10, 9, 8, 7), 4
- **Expected Outcome:** return 7
- **Actual outcome:** ✅

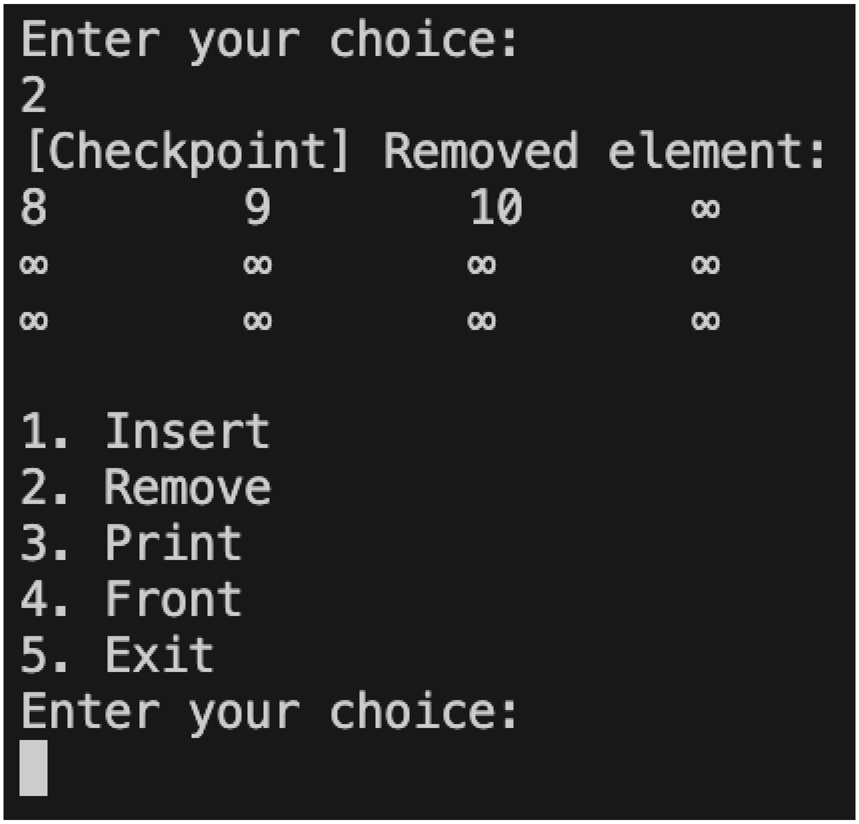


```
1. Insert
2. Remove
3. Print
4. Front
5. Exit
Enter your choice:
4
Front item: 7
```

Step 5 - Remove()

Test 1.1 - Remove and sort the heap matrix

- **Description:** Similar to insert method, except inverse BubbleDown and take the last (presumably lowest priority) value to compare with other values to swap and sort the matrix
- **Input: (**Pre-existing 3x4 matrix made up of numbers 10, 9, 8, 7), 2
- **Expected Outcome:** matrix printed in order or 8, 9, 10
- **Actual outcome:**✅

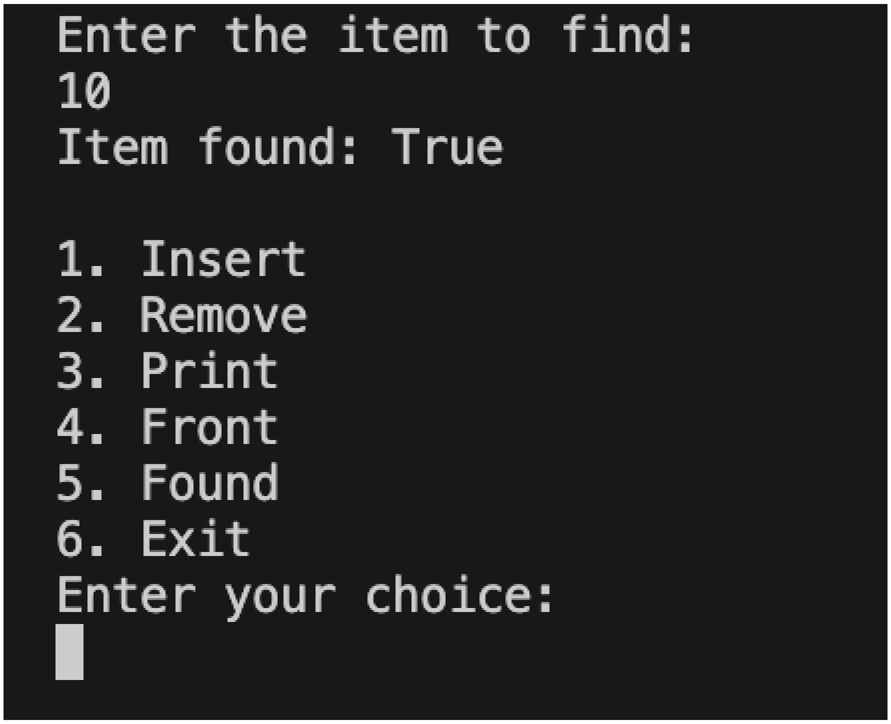


```
Enter your choice:
2
[Checkpoint] Removed element:
8       9       10      ∞
∞       ∞       ∞       ∞
∞       ∞       ∞       ∞

1. Insert
2. Remove
3. Print
4. Front
5. Exit
Enter your choice:
```

Step 6 - Found(int item)

Test 1.1 - Return true if item is found in Priority Queue

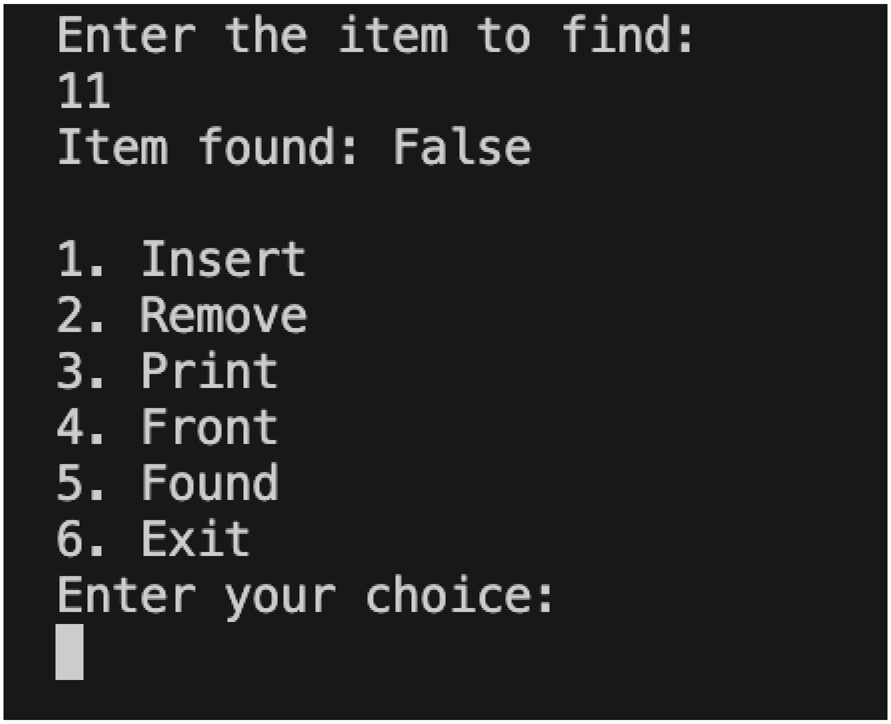
- **Description:** go through each row and column and compare each element at each position to see if it matches the item to be found
- **Input:** (Pre-Existing 3x4 matrix made up of numbers 10, 9, 8, 7), 5, 10
- **Expected Outcome:** Item found: True
- **Actual outcome:** ✅



```
Enter the item to find:
10
Item found: True

1. Insert
2. Remove
3. Print
4. Front
5. Found
6. Exit
Enter your choice:
```

Test 1.2 - Return False if item is not found in Priority Queue

- **Description:** go through each row and column and compare each element at each position to see if it matches the item to be found
- **Input:** (Pre-Existing 3x4 matrix made up of numbers 10, 9, 8, 7), 5, 11
- **Expected Outcome:** Item found: False
- **Actual outcome:** ✅

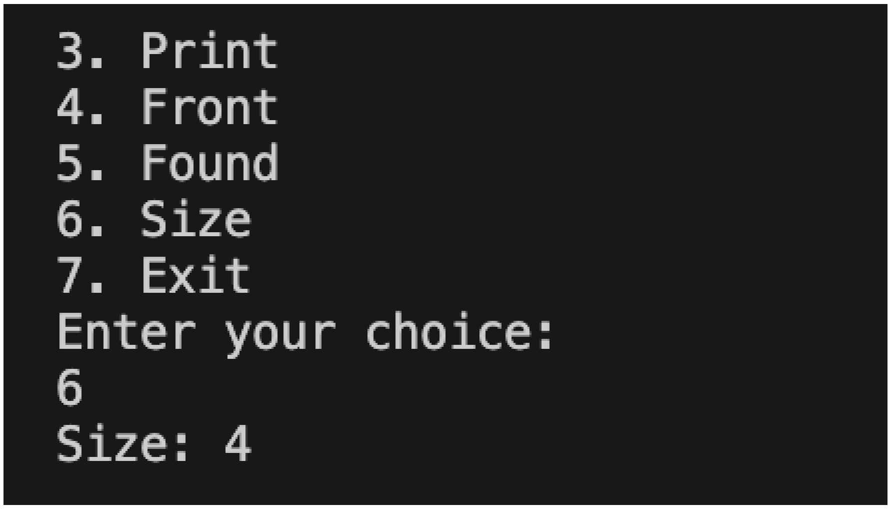


```
Enter the item to find:
11
Item found: False

1. Insert
2. Remove
3. Print
4. Front
5. Found
6. Exit
Enter your choice:
```

Step 7 - Size()

Test 1.1 - Return the size of the matrix

- **Description:** Return the amount of elements in the Priority Queue
- **Input:** (Pre-Existing 3x4 matrix made up of numbers 10, 9, 8, 7), 6
- **Expected Outcome:** Size: 4
- **Actual outcome:** ✅



```
3. Print
4. Front
5. Found
6. Size
7. Exit
Enter your choice:
6
Size: 4
```