Assignment 3 - Muzan Daffaallah

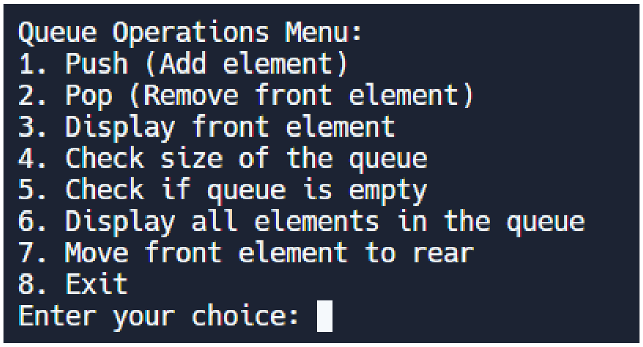
Part 1:

Running the code:

START. Create a new instance of Queue for integers. Display the menu with options. Get the user's choice from the menu. If the user chooses to push, ask for a number, call push with number, and print the success message. If the user chooses to pop, if the queue is not empty, call pop, print completion message. If the queue is empty, print error message. If the user chooses to display the front element, if the queue is not empty, call front and display the result. If the queue is empty print error message. If the user chooses to check if the queue is empty, call empty, if the queue is empty print "queue is empty" else print "queue not empty". If the user chooses to display all elements, call displayelements function, print all elements in queue. If the user chooses to move the front element to the rear, call move_to_rear function and print completion message. If the user chooses to exit the program, print exit message, break loop. Else if choice is invalid, print error message. **END.**

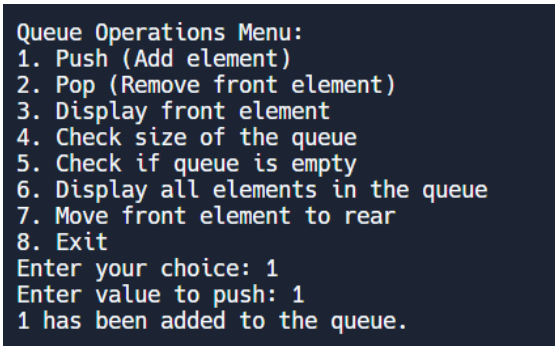
Screenshots:

Additions:
To help with the efficiency and usability of the program, i have added a user menu to make things easier for the user.



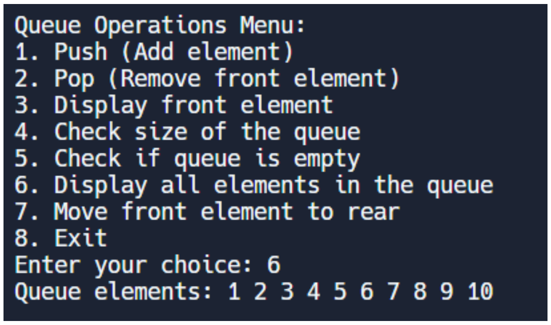
```
Queue Operations Menu:
1. Push (Add element)
2. Pop (Remove front element)
3. Display front element
4. Check size of the queue
5. Check if queue is empty
6. Display all elements in the queue
7. Move front element to rear
8. Exit
Enter your choice: 
```

Push:
If user chooses to push an element into the queue, they will be asked to provide an integer to push, and a completion message will be printed.



```
Queue Operations Menu:
1. Push (Add element)
2. Pop (Remove front element)
3. Display front element
4. Check size of the queue
5. Check if queue is empty
6. Display all elements in the queue
7. Move front element to rear
8. Exit
Enter your choice: 1
Enter value to push: 1
1 has been added to the queue.
```

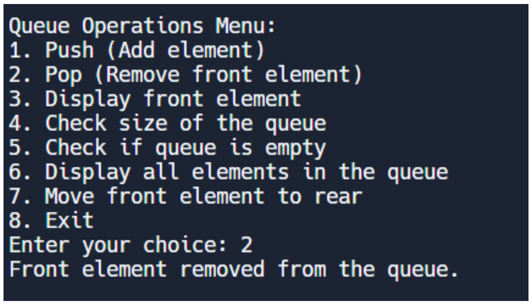
This is after 10 elements have been pushed:



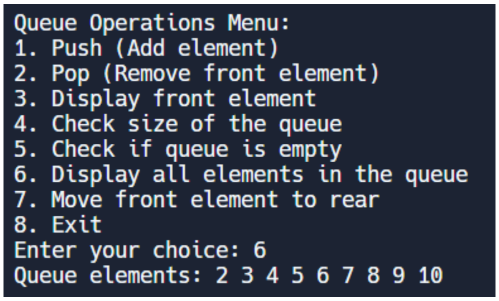
```
Queue Operations Menu:
1. Push (Add element)
2. Pop (Remove front element)
3. Display front element
4. Check size of the queue
5. Check if queue is empty
6. Display all elements in the queue
7. Move front element to rear
8. Exit
Enter your choice: 6
Queue elements: 1 2 3 4 5 6 7 8 9 10
```

Pop:

If a user chooses to pop an element, the element will be popped or if queue is empty, message will be printed.

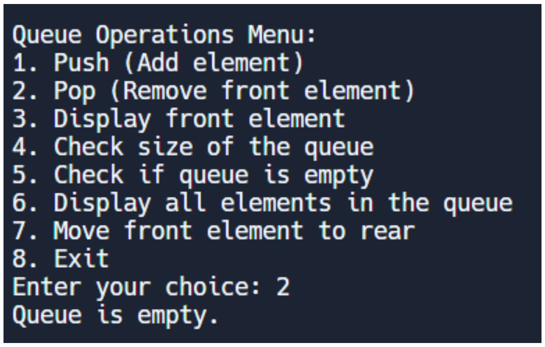


```
Queue Operations Menu:
1. Push (Add element)
2. Pop (Remove front element)
3. Display front element
4. Check size of the queue
5. Check if queue is empty
6. Display all elements in the queue
7. Move front element to rear
8. Exit
Enter your choice: 2
Front element removed from the queue.
```



```
Queue Operations Menu:
1. Push (Add element)
2. Pop (Remove front element)
3. Display front element
4. Check size of the queue
5. Check if queue is empty
6. Display all elements in the queue
7. Move front element to rear
8. Exit
Enter your choice: 6
Queue elements: 2 3 4 5 6 7 8 9 10
```

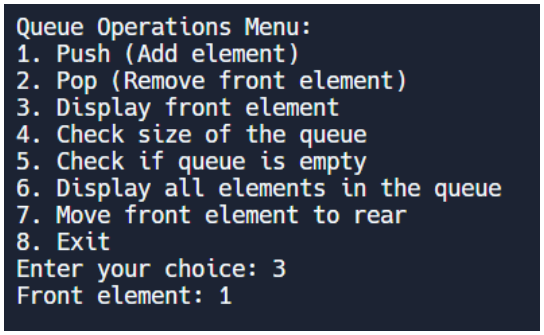
If empty:



```
Queue Operations Menu:
1. Push (Add element)
2. Pop (Remove front element)
3. Display front element
4. Check size of the queue
5. Check if queue is empty
6. Display all elements in the queue
7. Move front element to rear
8. Exit
Enter your choice: 2
Queue is empty.
```

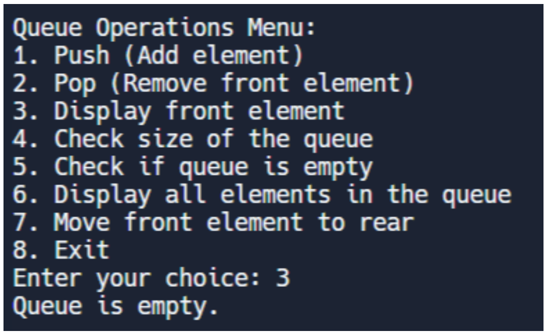
Front:

If the user chooses front, then the front element will be printed. If the queue is empty then it will print queue is empty message.



```
Queue Operations Menu:
1. Push (Add element)
2. Pop (Remove front element)
3. Display front element
4. Check size of the queue
5. Check if queue is empty
6. Display all elements in the queue
7. Move front element to rear
8. Exit
Enter your choice: 3
Front element: 1
```

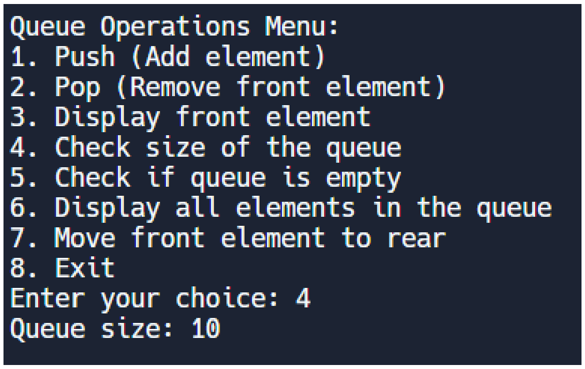
If empty:



```
Queue Operations Menu:
1. Push (Add element)
2. Pop (Remove front element)
3. Display front element
4. Check size of the queue
5. Check if queue is empty
6. Display all elements in the queue
7. Move front element to rear
8. Exit
Enter your choice: 3
Queue is empty.
```

Size:

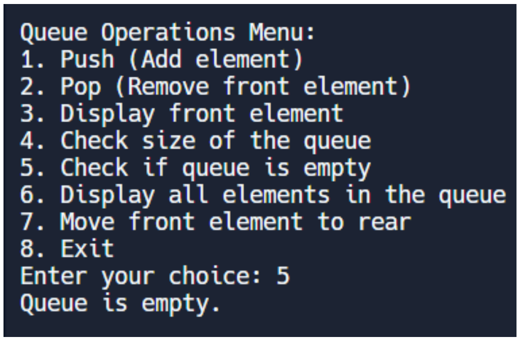
If the user chooses size, then the size of the queue will be printed.



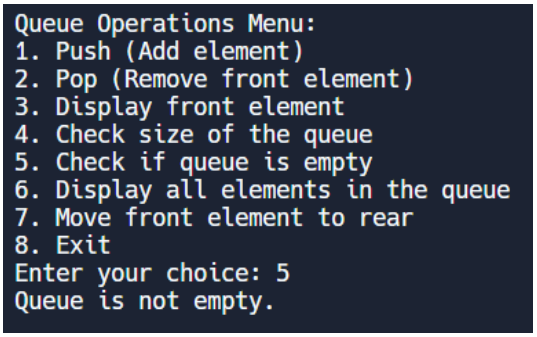
```
Queue Operations Menu:
1. Push (Add element)
2. Pop (Remove front element)
3. Display front element
4. Check size of the queue
5. Check if queue is empty
6. Display all elements in the queue
7. Move front element to rear
8. Exit
Enter your choice: 4
Queue size: 10
```

Empty:

If the users chooses to check if the queue is empty, it will either print “empty” if empty otherwise it will print “not empty”.



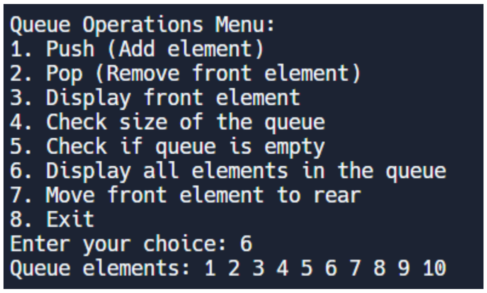
```
Queue Operations Menu:
1. Push (Add element)
2. Pop (Remove front element)
3. Display front element
4. Check size of the queue
5. Check if queue is empty
6. Display all elements in the queue
7. Move front element to rear
8. Exit
Enter your choice: 5
Queue is empty.
```



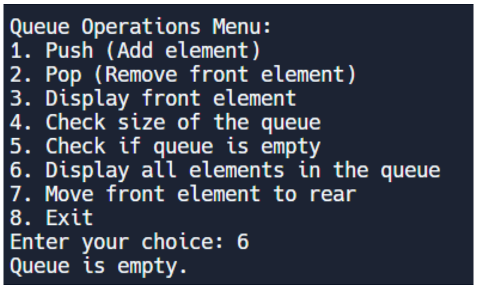
```
Queue Operations Menu:
1. Push (Add element)
2. Pop (Remove front element)
3. Display front element
4. Check size of the queue
5. Check if queue is empty
6. Display all elements in the queue
7. Move front element to rear
8. Exit
Enter your choice: 5
Queue is not empty.
```

Display Elements:

If the user chooses to display elements in queue it will print all elements, and prints message if no elements to be printed.



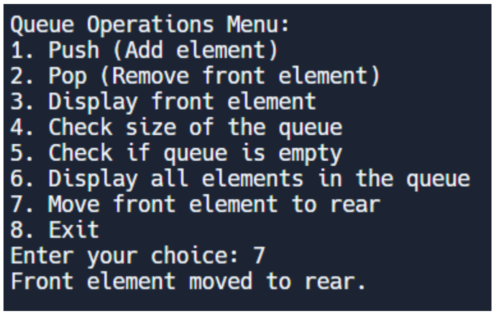
```
Queue Operations Menu:
1. Push (Add element)
2. Pop (Remove front element)
3. Display front element
4. Check size of the queue
5. Check if queue is empty
6. Display all elements in the queue
7. Move front element to rear
8. Exit
Enter your choice: 6
Queue elements: 1 2 3 4 5 6 7 8 9 10
```



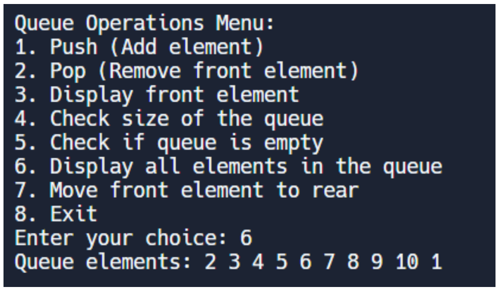
```
Queue Operations Menu:
1. Push (Add element)
2. Pop (Remove front element)
3. Display front element
4. Check size of the queue
5. Check if queue is empty
6. Display all elements in the queue
7. Move front element to rear
8. Exit
Enter your choice: 6
Queue is empty.
```

Move to Rear:

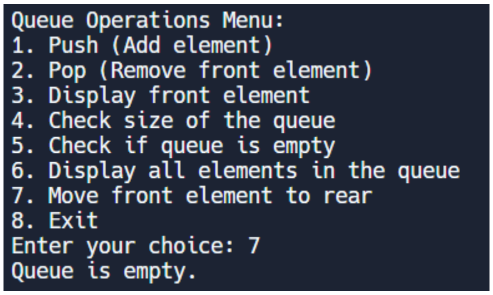
If the user chooses to move to rear the front element will be moved to the end of the queue and all elements will be shifted up 1. If queue is empty message will be printed.



```
Queue Operations Menu:
1. Push (Add element)
2. Pop (Remove front element)
3. Display front element
4. Check size of the queue
5. Check if queue is empty
6. Display all elements in the queue
7. Move front element to rear
8. Exit
Enter your choice: 7
Front element moved to rear.
```



```
Queue Operations Menu:
1. Push (Add element)
2. Pop (Remove front element)
3. Display front element
4. Check size of the queue
5. Check if queue is empty
6. Display all elements in the queue
7. Move front element to rear
8. Exit
Enter your choice: 6
Queue elements: 2 3 4 5 6 7 8 9 10 1
```



```
Queue Operations Menu:
1. Push (Add element)
2. Pop (Remove front element)
3. Display front element
4. Check size of the queue
5. Check if queue is empty
6. Display all elements in the queue
7. Move front element to rear
8. Exit
Enter your choice: 7
Queue is empty.
```

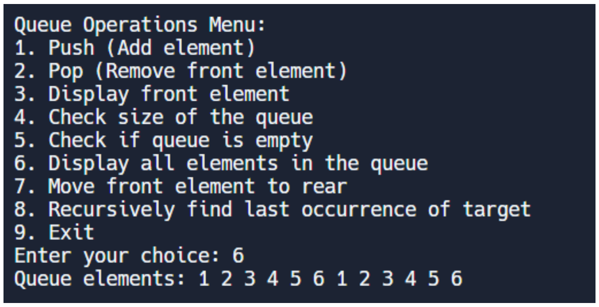
Part 2:

Running the code:

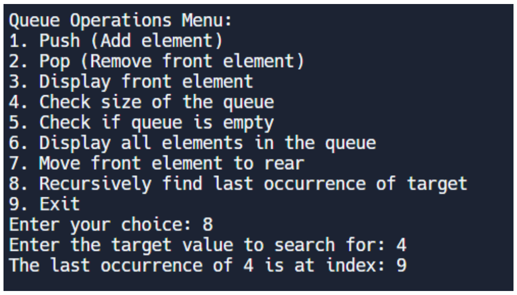
Addition to previous code. **START.** If user chooses to find the last occurrence of a target, then the program will recursively search through the queue. If the element is found, the index of the last occurrence of that element is printed. If that element is not in the queue, an error message is printed. And lastly if the queue is empty, an error message is printed. **END.**

Screenshots:

Find last occurrence of function:
User inputs target. If element is found, index is printed.

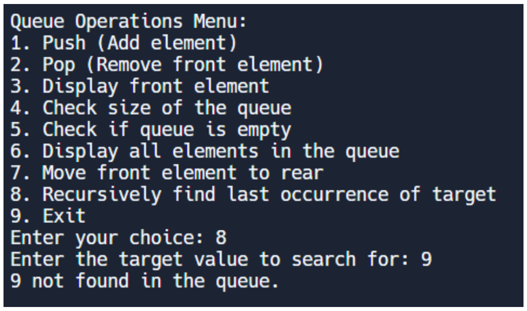


```
Queue Operations Menu:
1. Push (Add element)
2. Pop (Remove front element)
3. Display front element
4. Check size of the queue
5. Check if queue is empty
6. Display all elements in the queue
7. Move front element to rear
8. Recursively find last occurrence of target
9. Exit
Enter your choice: 6
Queue elements: 1 2 3 4 5 6 1 2 3 4 5 6
```



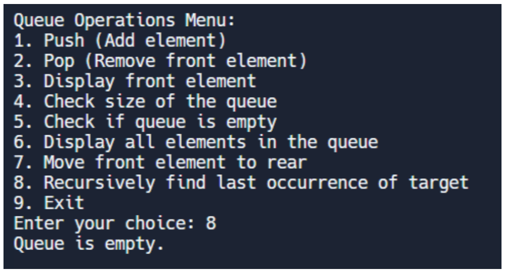
```
Queue Operations Menu:
1. Push (Add element)
2. Pop (Remove front element)
3. Display front element
4. Check size of the queue
5. Check if queue is empty
6. Display all elements in the queue
7. Move front element to rear
8. Recursively find last occurrence of target
9. Exit
Enter your choice: 8
Enter the target value to search for: 4
The last occurrence of 4 is at index: 9
```

If element is not found, message is printed.



```
Queue Operations Menu:
1. Push (Add element)
2. Pop (Remove front element)
3. Display front element
4. Check size of the queue
5. Check if queue is empty
6. Display all elements in the queue
7. Move front element to rear
8. Recursively find last occurrence of target
9. Exit
Enter your choice: 8
Enter the target value to search for: 9
9 not found in the queue.
```

If queue is empty, message is printed.



```
Queue Operations Menu:
1. Push (Add element)
2. Pop (Remove front element)
3. Display front element
4. Check size of the queue
5. Check if queue is empty
6. Display all elements in the queue
7. Move front element to rear
8. Recursively find last occurrence of target
9. Exit
Enter your choice: 8
Queue is empty.
```

Part 3:

Running the code:

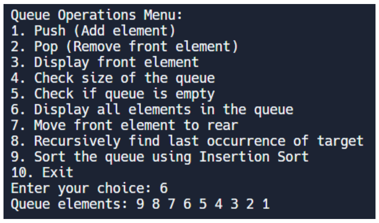
Addition to previous code (parts 1 & 2). **START.** If the user chooses to sort the queue, then insertion sort is used in order to sort it. If the queue is empty, a message is printed. **END.**

Screenshots:

Insertion Sort:

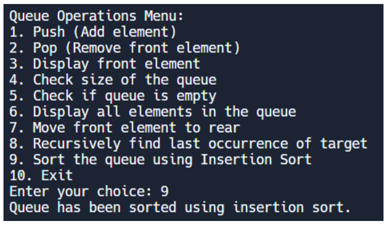
If the user chooses to sort the queue, then insertion sort is used in order to sort it. If the queue is empty, a message is printed.

Before:



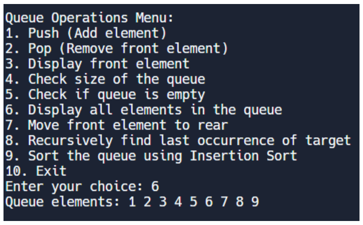
```
Queue Operations Menu:
1. Push (Add element)
2. Pop (Remove front element)
3. Display front element
4. Check size of the queue
5. Check if queue is empty
6. Display all elements in the queue
7. Move front element to rear
8. Recursively find last occurrence of target
9. Sort the queue using Insertion Sort
10. Exit
Enter your choice: 6
Queue elements: 9 8 7 6 5 4 3 2 1
```

During:



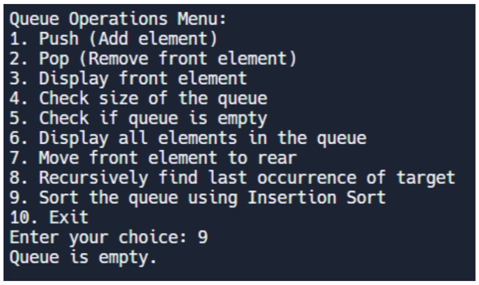
```
Queue Operations Menu:
1. Push (Add element)
2. Pop (Remove front element)
3. Display front element
4. Check size of the queue
5. Check if queue is empty
6. Display all elements in the queue
7. Move front element to rear
8. Recursively find last occurrence of target
9. Sort the queue using Insertion Sort
10. Exit
Enter your choice: 9
Queue has been sorted using insertion sort.
```

After:



```
Queue Operations Menu:
1. Push (Add element)
2. Pop (Remove front element)
3. Display front element
4. Check size of the queue
5. Check if queue is empty
6. Display all elements in the queue
7. Move front element to rear
8. Recursively find last occurrence of target
9. Sort the queue using Insertion Sort
10. Exit
Enter your choice: 6
Queue elements: 1 2 3 4 5 6 7 8 9
```

If empty:



```
Queue Operations Menu:
1. Push (Add element)
2. Pop (Remove front element)
3. Display front element
4. Check size of the queue
5. Check if queue is empty
6. Display all elements in the queue
7. Move front element to rear
8. Recursively find last occurrence of target
9. Sort the queue using Insertion Sort
10. Exit
Enter your choice: 9
Queue is empty.
```