Circular Queue Functions

Function	Code Snippet	Code Output

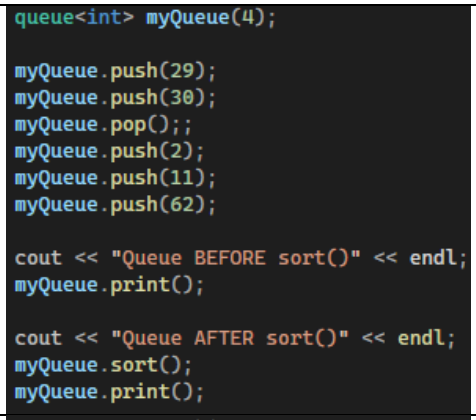
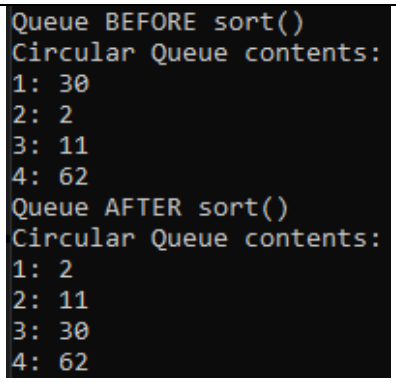
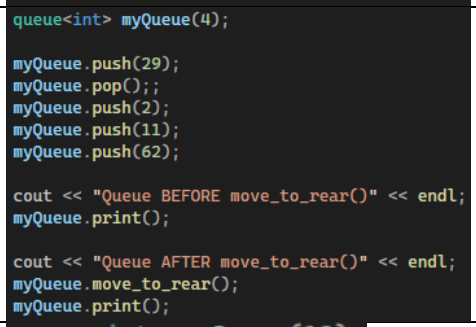
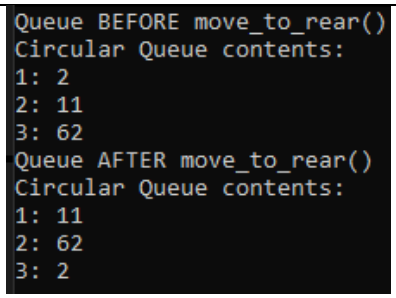
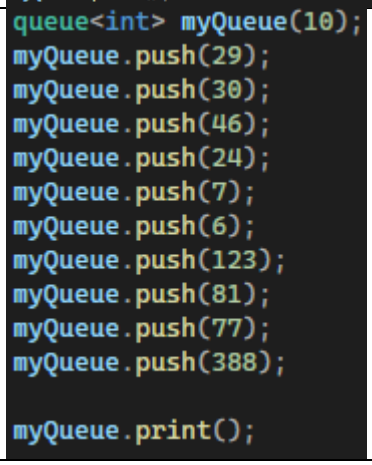
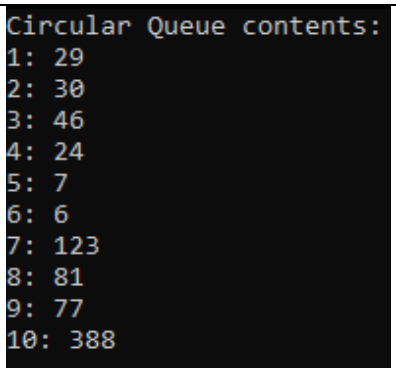
push: adds an element to the rear end of the queue	`queue<int> myQueue(4);`
	`myQueue.push(29);`
	`myQueue.push(30);`
	`myQueue.push(46);`
	`myQueue.push(2);`
	
	`myQueue.push(3);`
	`myQueue.push(4);`
	
	`myQueue.print();`
		`Queue is full`
		`Queue is full`
		`Circular Queue contents:`
		`1: 29`
		`2: 30`
		`3: 46`
		`4: 2`

Pop: removes an element from the front of the queu	`Before pop():`
	`Circular Queue contents:`
	`1: 29`
	`2: 30`
	`3: 46`
	`4: 2`
	`After pop():`
	`Circular Queue contents:`
	`1: 30`
	`2: 46`
	`3: 2`
		`queue<int> myQueue(4);`
		
		`myQueue.push(29);`
		`myQueue.push(30);`
		`myQueue.push(46);`
		`myQueue.push(2);`
		
		`cout << "Before pop():" << endl;`
		`myQueue.print();`
		
		`myQueue.pop();`
		
		`cout << "After pop():" << endl;`
		`myQueue.print();`

Front: Returns the item that's the 1st element of the queue	`queue<int> myQueue(4);`
	
	`myQueue.push(29);`
	`myQueue.push(30);`
	`myQueue.pop();`
	`myQueue.push(2);`
	`myQueue.push(11);`
	
	`myQueue.print();`
	
	`cout << "The front of the ";`
	`cout << " queue is " << myQueue.front() << endl;`
		`Circular Queue contents:`
		`1: 30`
		`2: 2`
		`3: 11`
		`The front of the  queue is 30`

size: Returns an integer that represents the number of items that are currently in the queue	`queue<int> myQueue(4);`
	
	`myQueue.push(29);`
	`myQueue.push(30);`
	`myQueue.push(2);`
	`myQueue.push(11);`
	
	`myQueue.print();`
	
	`cout << "The size of the ";`
	`cout << " queue is " << myQueue.size() << endl;`
		`Circular Queue contents:`
		`1: 29`
		`2: 30`
		`3: 2`
		`4: 11`
		`The size of the queue is 4`

empty: Returns a boolean that determines if the queue is empty or not
- True, if the queue has 0 items
	`queue<int> myQueue(4);`
	
	`if (myQueue.empty() == true) {`
	`    cout << "The Queue is empty" << endl;`
	`}`
	
	`myQueue.push(29);`
	`myQueue.push(30);`
	`myQueue.push(46);`
	`myQueue.push(2);`
	
	`if (myQueue.empty() == false) {`
	`    cout << "The Queue is not empty" << endl;`
	`}`
	
	`myQueue.print();`
		`The Queue is empty`
		`The Queue is not empty`
		`Circular Queue contents:`
		`1: 29`
		`2: 30`
		`3: 46`
		`4: 2`

<table>
<tr><td>- False, if the queue has at least 1 item</td><td></td><td></td></tr>
<tr><td>sort: Performs insertion sort on the queue and sorts the queue in ascending order</td><td> 
</td><td> 
</td></tr>
<tr><td>move_to_rear: Moves the index of the 1st element of the queue all the way to the rear end of the queue</td><td> 
</td><td> 
</td></tr>
<tr><td>print: Display all items of the queue in ascending order, from front of the queue to the back of the queue.</td><td> 
</td><td> 
</td></tr>
</table>

linear_search(): Finds an item in a vector and returns the index of the last occurrence of the item

```cpp
vector<int> myVector = { 3, 9, 6, 3, 22 , 1 , 3};

int index = linear_search(myVector, 3);

cout << endl;
if (index == -1) {
    cout << "Item not found" << endl;
}
else {
    cout << "Found " << myVector[index] << " at index " << index << endl;
}
```

```
Found 3 at index 6
```