1. (Weight: 10%) Explain why we need both an `iterator` and a `const_iterator`

2. (Weight: 10%) Indicate whether you should use an `iterator` or a `const_iterator` as a parameter in new functions for the list class that would perform each of these operations.
   a. Insert a new element after the current iterator position.
   b. Replace the data stored in the currently selected item.
   c. Retrieve the data stored in the currently selected item.
   d. Insert a new element before the currently selected item.

3. (Weight: 10%) **Programming:** Write a function that reads a line and reverses the words in the line (not the characters) using a stack. For example, given the following input:

```
The quick brown fox jumps over the lazy dog.
```

   You should get the following output:

```
dog. lazy the over jumps fox brown quick The
```

4. (Weight: 10%) Trace the evaluation of the following expression using class `Postfix_Evaluator`. Show the operand stack each time it is modified (Use `Postfix_Evaluator` on Blackboard).

```
10 2 * 5 / 6 2 5 * + –
```

   - Here is an example of tracing what happens when the first operand is parsed:

Expression	Action	Stack
10 2 * 5 / 6 2 5 * + – ↑	Push 10	10

   - Note: Take care – when you copy expressions (like the one above and the ones below) into visual studio. Make sure the minus sign (-) is copied as a minus sign not a dash. If it is copied as a dash, change it into a minus sign so that Visual Studio doesn't throw an error.

5. (Weight: 20%) Trace the conversion of the following expressions to postfix using the `Infix_To_Postfix` class. Show the operator stack each time it is modified. (You can find the class on Blackboard)

```
y – 7 * 35 + 4 / 6 – 10
( x + 15 ) * ( 3 * ( 4 – ( 5 + 7 / 2 ) ) )
```

6. (Weight: 10%) How would you modify class `Infix_To_Postfix` to handle the exponentiation operator, indicated by the symbol ^. The first operand is raised to the power indicated by the second operand. Assume that a sequence of ^ operators will not occur and that precedence('^') is greater than precedence('*').

7. (Weight:15%) Write an algorithm to display all the elements in a queue using just the queue operations. How would your algorithm change the queue?

8. (Weight:15%) **Programming:** Write a new queue function called move_to_rear that moves the element currently at the front of the queue to the rear of the queue. The element that was second in line will be the new front element. Do this using functions push, front, and pop.