1. Circular queues are used quite a bit in operating systems and high performance systems, especially when performance matters. Do a little outside research, and edit this section of the readme answering specifically: Why is a ring buffer useful and/or when should it be used?

Circular queues, also known as ring buffers, are widely used in high-performance systems and operating systems where performance is critical. A ring buffer is a type of queue that has its end connected to the beginning, forming a circular shape.

This data structure is highly useful because it provides efficient data storage and retrieval. It can handle large amounts of data without the need for dynamic memory allocation, which can negatively impact performance. In a ring buffer, data can be added and removed quickly without the need to move elements in memory, making it faster compared to conventional queues.

In real-time systems where data must be processed immediately, ring buffers are ideal. They allow for temporary storage of data without waiting for memory allocation, ensuring real-time processing.

Ring buffers are used in situations where data is produced and consumed at a high rate, such as audio/video playback, network communication, and real-time data processing. In these cases, a ring buffer offers fast and efficient data transfer, which is necessary for maintaining high performance.

2. We are going to talk about stacks quite a lot in this course, so it will be important to understand them. Do a little outside research, and edit this section of the README answering specifically: Why is a stack useful and/or when should it be used?

Stacks are a commonly used data structure in computer science and software engineering. They follow the last-in, first-out (LIFO) approach, which means that the most recent item added to the stack is the first one to be taken out.
There are several reasons why stacks are useful, including:
1. Managing function calls: Stacks are frequently used to keep track of function calls in a program, where the most recently called function is positioned at the top of the stack.
2. Reversing actions: Stacks can be utilized to reverse actions in applications, where the most recent action is placed at the top of the stack and can be reversed by removing it.
3. Checking symbol balance: Stacks can be used to verify balanced symbols in a program, such as ensuring all parentheses in an expression are correctly balanced.

4. Expressions calculation: Stacks can also be used to compute mathematical expressions by transforming the expression into postfix form and using a stack to evaluate it.

To summarize, stacks are a useful data structure when LIFO behavior is needed, such as for managing function calls, reversing actions, checking symbol balance, and computing expressions.