# Problem 262: Going in Circles

**Difficulty:** Medium

**Author:** Jonathan Tran, Grand Prairie, Texas, United States

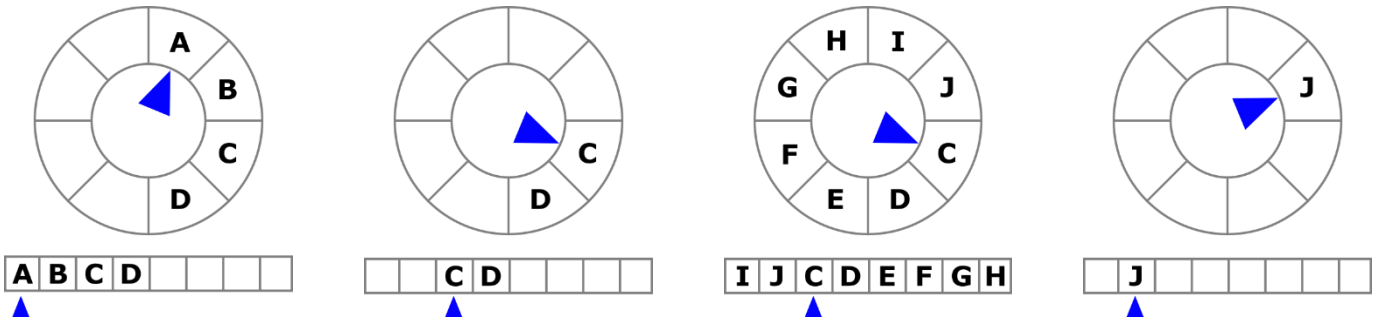
**Originally Published:** Code Quest 2025

## Problem Background

When you're watching a video on YouTube or some other website, your browser won't download the entire video at once; it'll only download a few seconds ahead of where you currently are. This process, called "buffering," allows the video to continue playing for a short period, even if your internet connection is briefly disrupted. Video buffering frequently makes use of a data structure called a "circular buffer."

Circular buffers have a fixed size and store data in a circular (wrap-around) pattern. These are often used where memory is limited, and where dynamically allocating memory (increasing memory usage as needed) isn't ideal. Circular buffers work by filling a pre-allocated array with data as it arrives. A memory pointer keeps track of the "start" of the buffer; even though a circle doesn't have a start point, the computer has to know where to begin reading data in the buffer when needed. As data is read from the buffer, it is removed, freeing up space for more data.

See the example below; in this diagram, the characters A B C D are stored into a circular buffer with a size of 8. Two of these are then read and removed: A and B. The "start" pointer is then moved ahead to the next unread character, C. Next, the characters E F G H I J are stored; I and J wrap around to get placed in A and B's now-vacant positions. Seven characters are then read; the pointer marking the "start" of the buffer also wraps around when it reaches the end of the array.




## Problem Description

Your team is working on implementing a circular buffer for use with a VR flight simulator being built with Lockheed Martin's Rotary and Mission Systems. The quality control team wants some tests built to demonstrate that the buffer algorithm works as intended. You need to create a program that can accept the following commands (which will be used in a script provided by the quality control team) and properly process them:

- ADD - This command requires your program to add one or more characters to the buffer, starting at the first empty position after the read position. If the buffer is or becomes full, existing data should be overwritten in a circular manner, and the read position should be advanced to the first position that was not overwritten.
- CONSUME - This command requires your program to remove the specified number of characters from the buffer, starting with the character at the current read position. The read position should be advanced to the position after the last character removed. If the number of characters to consume exceeds the number actually stored in the buffer, all characters are consumed and the buffer will be emptied.
- SHOW - This command requires your program to print some information about the current state of the buffer. If the buffer contains an odd number of characters, your program should print the character located at the midpoint between the current read position and the last character within the buffer. If the buffer contains an even number of characters, your program should print the two characters to either side of that midpoint. If the buffer is empty, your program should print "Empty". This command does not add or remove data from the buffer, and does not change the read position.

## Sample Input

The first line of your program's input, **received from the standard input channel**, will contain a positive integer representing the number of test cases. Each test case will include the following lines:

- A line containing two positive integers separated by a space:
  - N, representing the number of commands included in this test case
  - S, representing the size of the circular buffer.
- N lines each containing a single command and its arguments. The command and each argument (if any) will be separated by spaces. Possible commands and arguments include:
  - ADD, which will be followed by one or more visible and printable ASCII characters (see the US ASCII table in the reference information for examples; spaces will not be given as arguments)
  - CONSUME, which will be followed by a single non-negative integer
  - SHOW, which has no arguments

```
2
4 5
ADD A B C
SHOW
ADD D
SHOW
9 6
ADD a b c d
CONSUME 1
SHOW
CONSUME 4
SHOW
ADD e f g h i
SHOW
ADD j k l
SHOW
```

## Sample Output

For each test case, your program must print the output from any SHOW commands received during the test case. The result of each command should be printed on a separate line; when two characters must be printed, print them in the order read and separated by a single space.

```
B
B C
c
Empty
g
i j
```