Table of Contents

[Solutions 1](#_Toc178809542)

[1 Ping Pong Question 1](#_Toc178809543)

[1.1 Original Code 1](#_Toc178809544)

[1.2 Output 2](#_Toc178809545)

[2 The desired output through Step 1 2](#_Toc178809546)

[2.1 Problem Statement - Step 1 2](#_Toc178809547)

[2.2 Edited code for Step 1 3](#_Toc178809548)

[2.3 Output 4](#_Toc178809549)

[3 The desired output through Step 2 4](#_Toc178809550)

[3.1 Problem Statement - Step 2 4](#_Toc178809551)

[3.2 Edited Code for Step 2 4](#_Toc178809552)

[3.3 Output 5](#_Toc178809553)

[4 Explanation 6](#_Toc178809554)

**List of Code Snippets**

[Code 1: Given Code from the question. 1](#_Toc178809582)

[Code 2: Edit to the code made Problem Statement - Step 1 instructions. 3](#_Toc178809583)

[Code 3: Edited code after Problem Statement - Step 2 instruction. 5](#_Toc178809584)

**List of Figures**

[Figure 1: Raw un-edited program output. 2](#_Toc178809633)

[Figure 2: First edit to the program based on Problem Statement - Step 1 instructions. 4](#_Toc178809634)

[Figure 3: Final edit to the program based on Problem Statement - Step 2 instructions. 6](#_Toc178809635)

Solutions

# Ping Pong Question

Consider the following Go code where two goroutines ping and pong are supposed  
to output “ping” and “pong” alternatively by collaborating via channels.

## Original Code



Code 1: Given Code from the question.

The above Code 1 represents the original code given in the problem statement.

## Output

The immediate output derived from running the raw program given in the problem statement is as given below in the Figure 1. It is observed that the program doesn’t behave as intended and doesn’t produce the necessary expected output.

A screenshot of a computer

Description automatically generated

Figure 1: Raw un-edited program output.

# The desired output through Step 1

## Problem Statement - Step 1

The desired output should be

Start!  
ping 0

pong 0

ping 1

pong 1

ping 2

pong 2

ping 3

pong 3

ping 4

pong 4

ping 5

pong 5

ping 6

pong 6

ping 7

pong 7

ping 8

pong 8

ping 9

pong 9

However, go reports a deadlock after displaying “Start!”. Add a line at line A  
so that ping and pong will start to output messages as desired. (Hint: send  
something to ping!)

## Edited code for Step 1



Code 2: Edit to the code made Problem Statement - Step 1 instructions.

The code snippet present above in Code 2 represents the edits made to the original code which fixed part of the problem as suggested by the problem statement.

## Output

The edited output from the first step of the process in editing the original program given in the problem statement is as given below in the Figure 2. It is observed that the program behaves as per the expectations set by The desired output through Step 1.

A screenshot of a computer

Description automatically generated

Figure 2: First edit to the program based on Problem Statement - Step 1 instructions.

# The desired output through Step 2

## Problem Statement - Step 2

Still, go reports a deadlock after displaying all desired messages. Modify  
line B to resolve the issue. Explain why. (Hint: what if you send something to a  
channel but no one is receiving?)

## Edited Code for Step 2

The code snippet present above in Code 3 represents the secondary edits made to the Edited code for Step 1 which fixed the entire problem as suggested by the problem statement.



Code 3: Edited code after Problem Statement - Step 2 instruction.

## Output

The edited output from the final step of the process in editing the original program given in the problem statement is as given below in the Figure 3. It is observed that the program behaves thoroughly as per the expectations set by The desired output through Step 2, thus solving the entire problem.

A screen shot of a computer

Description automatically generated

Figure 3: Final edit to the program based on Problem Statement - Step 2 instructions.

# Explanation

The pong go-routine tries to broadcast a value on the pi channel (out <- false) after the final pong message is displayed, but there is no recipient for this value, which results in a deadlock. This is because the out channel (po) has already been closed by the ping go-routine after completing its iterations.

The pong go-routine may communicate its final value without blocking, even in the absence of a recipient, by creating a buffered channel with a capacity of 1, which we call pi. This breaks the impasse.

A predetermined number of values can be stored on a buffered channel before the transmit process stops. In this instance, a buffer size of 1 is plenty to keep the intended behavior in place and avoid the deadlock.