3.1. When TEST3a is 0, main3 does not spawn the second process, causing the process to execute normally after waking up from sleepms, outputting “Rested and ready” repeatedly, as shown in Figure 1. On the other hand, when TEST3a is 1, wrongturn175 process is executed, overwriting the traveler's saved return address with the address of delphi(), outputting “Welcome to Delphi”, as shown in Figure 2. Thus, we have verified that wrongturn175() achieved its objective.

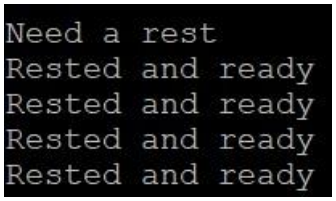

Figure 1

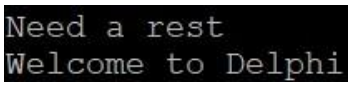

Figure 2

3.2. The test results are the same as the 3.1. When TEST3a is 0, main3 does not spawn the second process, causing the process to execute normally after waking up from sleepms, outputting “Rested and ready” repeatedly, as shown in Figure 3. When TEST3a is one, the process jumps to delphi() and outputs “Welcome to Delphi”, as shown in Figure 4. Hence, we have verified that the detour works correctly.

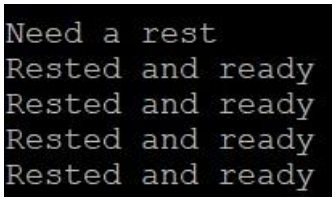

Figure 3

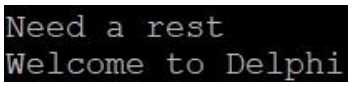

Figure 4

4. In the one-sender test, we successfully started the receiver and sender, registering the callback function, and returned control back to the receiver. See the test result in Figure 5. In the two-senders test, we successfully sent and received the second message, as shown in Figure 6. Thus, we have verified that our implementations are correct.

```
Send 2 4 2
Receiver (PID 4) registered callback
Send 5 307 22
Sender (PID 5) sent message 22 to PID 4
receive 4 740 22
Receiver (PID 4) still running, last msg: 22
Receiver (PID 4) still running, last msg: 22
Receiver (PID 4) still running, last msg: 22
Receiver (PID 4) still running, last msg: 22
Receiver (PID 4) still running, last msg: 22
Receiver (PID 4) still running, last msg: 22
Receiver (PID 4) still running, last msg: 22
```

Figure 5

```
Send 2 4 2
Receiver (PID 4) registered callback
Send 5 307 22
Sender (PID 5) sent message 22 to PID 4
receive 4 739 22
Receiver (PID 4) still running, last msg: 22
Receiver (PID 4) still running, last msg: 22
Send 6 1306 44
Sender (Preceive 4 1307 44
ID 6) sent message 44 to PID 4
Receiver (PID 4) still running, last msg: 44
Receiver (PID 4) still running, last msg: 44
Receiver (PID 4) still running, last msg: 44
Receiver (PID 4) still running, last msg: 44
Receiver (PID 4) still running, last msg: 44
Receiver (PID 4) still running, last msg: 44
```

Figure 6

Bonus. In this test, we successfully printed “Welcome to Delphi” and returned to needarest to continue execution. The results are shown in Figure 7. The shortcoming of this implementation is that it relies on manually inspected offsets without robust error handling, making it difficult to debug if something went wrong.

```
Need a rest
Welcome to Delphi
Rested and ready
Rested and ready
Rested and ready
Rested and ready
Rested and ready
```

Figure 7