To ensure that blank lines are not entered, we can implement a helper function `setStdin` as much as the example below:

*void* setStdin(*const char*\* input) {  
 *int* fds[2];  
 pipe(fds);  
 write(fds[1], input, strlen(input));  
 write(fds[1], "\n", 1);  
 close(fds[1]);  
 dup2(fds[0], STDIN\_FILENO);  
 close(fds[0]);  
}

This helper function redirects standard input to pipe. That can simulate the user input and can include blank line for testing valid input. Also, I use gTest assertions EXPECT\_STREQ to validate the blank lines as the code below:

TEST\_F(InputTest, WhiteBoxTestInputCustomerWithRetry) {  
 *struct* Customer customer;  
  
 setStdin("John\nSmith\n25 Elm St.\nToronto\nON\n\nM2E 4X4");  
 inputCustomer(&customer);  
 EXPECT\_STREQ(customer.firstName, "John");  
 EXPECT\_STREQ(customer.lastName, "Smith");  
 EXPECT\_STREQ(customer.streetAddress, "25 Elm St.");  
 EXPECT\_STREQ(customer.city, "Toronto");  
 EXPECT\_STREQ(customer.province, "ON");  
 EXPECT\_STREQ(customer.postalCode, "M2E 4X4");  
}

Testing that the correct prompt is issued for blank lines, we could capture the standard output and compare it with the expected output. We can redirect standard output to a custom buffer during the test to verify the prompt is correctly generated. First, we capture standard output during the unit test. Next, we invoke the function to test. During the test, the prompt from the function will be redirected to the buffer we had created. Finally, we compare the captured output with the expected prompt message to ensure they are correct.