Modifying the Z bit

Compile password.cc, then run the executable. This can be done on whatever OS you are currently using. However, to reverse engineer it, you will need to compile it on Windows. Why? Well, because the **free version** of the tool that we will be using to reverse engineer (IDA Pro) only works on Windows. Meh. Anyways, you won't guess the password. Well, at least not unless you took a look at the source code!

So how do we break the password? We can reverse engineer! We can use the IDA Pro disassembler to workaround this. Download the freeware version:

https://www.hex-rays.com/products/ida/support/download_freeware.shtml

To reverse engineer:

(1) Launch IDA Pro
(2) Select Go to "work on your own"
(3) Open the executable in IDA Pro: File | Open | password.exe | OK
(4) Wait until the initial autoanalysis is finished
(5) Search for text (via Alt+T) and search for the string "password:"
(6) Select the first occurrence of the string
(7) Note the flow of the program (especially the branch that leads to SUCCESS or FAILURE), and set a breakpoint at the jump (JZ) instruction immediately before this (via F2)
(8) Run the debugger (via F9) to show the program running with an incorrect password (try "password")
(9) The debugger will pause once you enter the password, allowing you to explore the flowchart and see which branch the program will take (FAILURE)
(10) Continue debugging past the breakpoint (via F9) to end the program
(11) To prevent the window from disappearing so quickly so that we can see the result of password entry, we can set another breakpoint somewhere near the end (e.g., after SUCCESS or FAILURE is displayed)
(12) Run the debugger again and provide an incorrect password
(13) Run the debugger again, but this time provide the correct password: "CHEEZ-IT GROOVES crispy cracker chips" and note SUCCESS
(14) Run the debugger one last time, providing an incorrect password, stop at the "flashing" branch to SUCCESS or FAILURE, and note the Z bit (ZF) value at the breakpoint (located in the General Registers window)
(15) Change the Z bit value from 1 to 0, and note the "flashing" branch change
(16) Continue the debugger (via F9) to SUCCESS (even with an incorrect password!)

So we can change things while in IDA Pro to alter the behavior of the executable. In this case, we could proceed beyond the password prompt to whatever would normally be there, without actually having to enter the correct password!

It would be useful if we could patch the executable so that we could run it normally and enter in any password when prompted. We'll discuss this later.

Note: in case you are interested in compiling the source code on Linux to a Windows executable (you will still need to disassemble the executable using IDA Pro on Windows):

```
sudo apt-get install mingw32
/usr/bin/i586-mingw32msvc-g++ password.cc -o password.exe
```