Buffer overflow

You are given a Linux program called grade that runs on the Rutgers iLab Linux systems. Download the file here and unzip it.

When you run it, it prompts you with

```
Do you want your grade (y/n)?
```

If you enter a y, you will see your grade, which is an F.

Fortunately, the code for this program is sloppy and it fails to check for the end of the buffer when reading the user's response. You need to take advantage of this to provide an input that will display your grade as an A+.

Groups

This is an individual assignment. You are expected to do this on your own and submit your own version of the injection and program output.

Environment

Your submissions will be tested on Rutgers iLab Linux systems. You can most likely develop this on any other x64 architecture Linux system but you are responsible for making sure that it will work on the iLab systems.

What you need to do

This grade change is possible via an overflow since it happens that the grade string (the symbol `grade`) is located in higher memory than the string containing the user's answer to the prompt (the symbol `answer`).

The caveat is that there is data in between the string that should not be modified.

You will need to use the gdb debugger to set a breakpoint prior to reading the input at a function called getinput() and see what the contents are in between the two buffers so you can recreate that data in your buffer overflow.

The data that you feed the program will be the response string (y) followed by the recreated data, followed by the string "A+", which represents your new grade.

Since you cannot type in most of this data, you will create a file containing the hex values for all the data as text (e.g., 0x11 0x22 0x33 …) and then use the spitbytes program to read the file and pipe its output to the grade program:

```
./spitbytes bytes.txt | ./grade
```

The spitbytes program simply reads whitespace-separated numbers that are represented in printable ASCII text (e.g., `42` for decimal, `0x2a`in hex) and outputs them in binary.

Hints

There is absolutely no programming that needs to be done here. Some of you will finish this assignment in just a few minues. Some of you, however, may not have any experience with using

gdb and may feel a bit lost in dumping memory and constructing the sequence of bytes you need to inject. Allow yourself sufficient time and realize that we have an exam coming up.

References

There are many tutorials on using gdb on the web. Here's one but you can find hundreds of others. All you need to know are:

- the `break` command
- the `run` command
- the `x` command to examine memory
- finding addresses that correspond to symbols (either the `print` or `info` commands or just dump contents and see the address via the `x` command.

You'll also need to know the ASCII codes for the charactes you need to enter and you can easily find these on the web (e.g., wikipedia) or run man ascii on any Linux or macOS system.

What to submit

When you have completed your assignment, you will need to submit only two files via sakai:

1. `bytes.txt`, which is a file that contains the textual representations of the data that you will send as input to the grade program.
2. `output.txt`, which contains the exact output of everything printed by running the grade program. For example:

```
./spitbytes bytes.txt | ./grade >output.txt
```

Validate that the contents are correct before submitting. We will use a script to generate and test your programs, and you will be penalized for any deviation. Do not submit anything else.