Assignment 10 – Stack smashing and feedback

Question 1: Stack smashing (8 points)

Read the following code and answer questions below.

```c
#include "stdio.h"
#include "string.h"

void buggy_print(char *msg, int len) {
    char buf[16];
    memcpy(buf, msg, len);
    printf("msg is %s!\n", buf);
}

void attack() {
    printf("you've been attacked!\n");
}

int main() {
    buggy_print("hello", sizeof("hello"));
    int num = /* TODO: fill in an integer */;
    void *ptrs[num];
    for (int i=0; i<num; i++) {
        ptrs[i] = attack;
    }

    buggy_print((char*) ptrs, sizeof(void *)*num);
}
```

1.a (3 points) Save the above code to a file, "demo.c".
Fill in "TODO" with an integer 1.
Compile and run the code:

```
$ gcc -o demo demo.c -fno-stack-protector -z execstack
$ ./demo
```

What do you see as the result and why?
Write down the output of the program and explain why this happens in 2-3 sentences.

1.b (3 points)

Now, fill in TODO with an integer 10.
Compile and run the code:

```
$ gcc -o demo demo.c -fno-stack-protector -z execstack
$ ./demo
```

What do you see as the result and why?
Write down the output of the program and explain why this happens in 2-3 sentences.

hints:
- recall x86 instruction “call” and “ret”; what do they do?
- in particular, where the “$rip” (instruction pointer) will point to after “ret”.

1.c (2 points) In the above execution (when “int num=10”), does the program crash (namely, segmentation fault)? If so, why?
If there is a segmentation fault, explain why in 2-3 sentences.
If your program quits normally, write “None”.

Question 2: Feedback (2 points)

This is to gather feedback. Any answer, except a blank one, will get points.

2.a (1 point)
Please state the topic or topics in this class that have been least clear to you.

2.b (1 point)
Please state the topic or topics in this class that have been most clear to you.

2.c (optional) anything you'd like us to know?