Problem Set 2: Looping, Arrays, Strings

Please send back via NYU Classes

- A zip archive named as
  `PS02_<your name as FirstLast>.zip`
  containing all C code files and answers to all questions.

For each problem of this set, you are asked to write a C program and answer some questions.

Total points: 90

Problem 1 (10 points): lexical scoping

`scope1.c, scope2.c`

1. Program `scope1.c`. In your `main()` function, put the following code:

```
int i;
for (i = 1; i <= 10; i++) {
    printf("%d\n", i);
}
printf("%d\n", i);
```

   a. What does it print on the last line?
   b. Why does it print a number that you did not specify?

2. Program `scope2.c` as above. This time, however, replace the second line with the following code (notice the added `int`):

```
for (int i = 1; i <= 10; i++) {
```

   a. What does it print on the last line this time?
   b. Is this number different? Why (or why not)? What does adding the `int` within the loop initialization do?

Problem 2 (20 points): buffer overrun and `strncpy()`

`buf1.c, buf2.c, buf3.c, buf4.c`

Buffer overrun, or buffer overflow, refers to the situation where an array is accessed at an index that is out of bounds, and is a typical example of “illegal” code. This problem uses variations of an example C program to show how buffer overrun can happen using the `strcpy()` function.

1. Program `buf1.c`. The `strcpy()` function accepts two strings and copies the content

of the second to the first. Write `buf1.c` to verify that the program runs as expected.

```
#define _FORTIFY_SOURCE 0     // turns off some warnings
#include <stdio.h>
#include <string.h>

int main(void) {
    char src[24] = "C programming";
    char dest[16];

    strcpy(dest, src);

    printf("dest: %s\n", dest);
    printf("src : %s\n", src);

    return 0;
}
```

2. Program `buf2.c` This time, change the variable `src` to have string `"C programming is so fun"`. Name this program `buf2.c` and test if the program runs as expected.
    a. What happens when you execute `buf2.c`?
    b. Why do the output of `dest` and `src` appear this way? (This is tricky.)
    c. How can you modify the program so that `dest` and `src` have the same string?

3. Program `buf3.c`. Function `strncpy()` is a function designed to accept a maximum number of bytes to copy. Copy `buf2.c` to create a new program `buf3.c` and in `buf3`.c replace the line calling `strcpy()` with the following code, which passes the size of `dest` to prevent buffer overrun:

```
strncpy(dest, src, sizeof(dest));
```

    a. Was `dest` printed as a string of length 16? If not, explain why.

4. Program `buf4.c`.Because of the NULL terminator, a `char` array of length 16 can hold a string that has up to 15 characters. Copy your `buf3.c` to a new program `buf4.c` and modify that to make it print `dest` as `"C Programming i"` which has exactly 15 characters. Hint: Do something to modify `dest[15]`, the last element of the array.

Problem 3 (10 points): buffer overrun and `scanf()`

`buf5.c, buf6.c`
Program `buf5.c`.Take the program `buf1.c` from the previous problem and replace the `strcpy()` line with the following line, which will make the program accept a single word from the keyboard input, and store it in `dest`:

```
scanf("%s", dest);
```

a. Run the program and type "12345678901234567890" to see if `src` is changed. Why?
b. Copy `buf5.c` into a new program `buf6.c`. Modify `scanf()` in `buf6.c` to use the format string "%15s", and see how `src` and `dest` look now. Can you explain what caused the difference?

Problem 4 (30 points): Pascal's Triangle and multi-dimensional arrays

```
          1
        1   1
      1   2   1
    1   3   3   1
  1   4   6   4   1
1   5  10  10   5   1
```

`pascal1.c, pascal2.c`
This is Pascal's Triangle, where the leftmost and rightmost entries of each row are 1 and the others are the sum of the two numbers above each. We are going to make a program that prints a skewed version of Pascal's Triangle:

```
1
1   1
1   2   1
1   3   3   1
(... etc)
```

In this problem, we will build a Pascal's Triangle again, using a two-dimensional array:

```
int pascal[16][16];
```

You can now build nested for-loops that calculate the numbers using:

```
pascal[i][j] = pascal[i-1][j-1] + pascal[i-1][j];
```

In order to align the numbers, use format string "%5d" to print the number with padded spaces.

1. Program `pascal1.c`. Write a program that uses nested for-loops to print Pascal's triangle from row 0 to 15. The numbers should be aligned as above.

2. Program `pascal2.c`. Instead of printing numbers, modify your program to print "*" if the number is odd, and " ", a space, if even. Print row 0 to 31.
    a. What patterns, if any, do you see?

HINTS:

- The first `for`-loop will iterate over the row index `i`, starting from `0`.
- The second loop index will iterate over the column index `j`, from `0` to `i`.
- Check to see if i is 0 or j, and in this case initialize to 1.
- In (2), make use of the modulus operator the conditional operator.

Problem 5 (20 points): word splitter with `strtok()`

`strtok.c`

The following code implements a simple command line interface, where the typed command is just repeated back to the user, until the user types an empty line to finish.

```
#include <stdio.h>
#include <string.h>
#include <stdbool.h>

int main(void) {
    char line[512] = {0,};

    while (true) {
       printf("Enter a line of text:\n");
       fgets(line, sizeof(line), stdin);

        if (line[0] == '\n') {
            puts("bye");
            break;
        }

        printf("You typed: %s\n", line);
    }

    return 0;
}
```

Using the above code as an example, write a program `strtok.c` to print one word on each line. For the input “C programming is so fun”, the output should be:

```
You typed:
C
programming
is
so
fun
```

Hint: You will need to replace `printf()` with multiple calls to `strtok()`.