Lab 10

Instructions: Implement a function to randomly sort an array of integers.

Objectives:

- Continued practice with integer arrays.
- Continued practice with functions/function calls.
- Continued practice with loops.

Task 1: Download the source file Lab10a.c. In this file, you will fill in blank functions to implement a calculator for rectangle perimeter and area, along with a function to print the attributes of a given rectangle (namely length, width, perimeter, and area).

Start by defining a struct to represent a rectangle. This struct should have two fields:

`rectangle.length`	Length of rectangle (integer)
`rectangle.width`	Width of rectangle (integer)

Once you've defined your structure, fill in the following functions to the given specifications:

- `getPerimeter`: Calculates the perimeter of a given rectangle r. (2*length+2*width)
- `getArea`: Calculates the area of a given rectangle r. (length*width)
- `printRect`: Prints the attributes of a given rectangle as in the following example:
  `length = 1, width = 2, area = 2, perimeter = 6`

Do not modify the `main` function. Once the above functions are correctly filled in, the program should run as follows:

```
/home/user/CIS190/Lab10$ ./Lab10a.out
Creating r = 4 x 8...
getPerimeter(r) = 24
getArea(r) = 32

Creating rectArr[0] = 0 x 0: length = 0, width = 0, area = 0, perimeter = 0
Creating rectArr[1] = 1 x 2: length = 1, width = 2, area = 2, perimeter = 6
Creating rectArr[2] = 2 x 4: length = 2, width = 4, area = 8, perimeter = 12
Creating rectArr[3] = 3 x 6: length = 3, width = 6, area = 18, perimeter = 18
Creating rectArr[4] = 4 x 8: length = 4, width = 8, area = 32, perimeter = 24
```

Figure 1. Correct output for `Lab10a.c`.

Task 2: Download the source file Lab10b.c. In this file, you will fill in blank functions to implement a stack of integers. A stack is a first-in-last-out data structure which allows us to access only the last element inserted, in the same way we can only grab the top plate of a real stack of plates.

Start by defining a struct to represent a stack. In C, we can represent a stack using two fields:

`stack.arr`	Array which holds the integers on the stack. Has a length of `STACK_CAP` elements, but does not necessarily use all of the spots.
`stack.n`	Keeps track of how many integers are currently on the stack, and therefore how many spots in `stack.arr` are currently being used. `stack.n == 0` → stack is empty; `stack.n ==` `STACK_CAP` → stack is full.

Once you've defined your structure, fill in the following functions to the given specifications.

- `push`: Places a new integer at `stack.arr[stack.n]`, then increments `stack.n` to update how many spots are being used. **If the stack is full, returns -1.**
- `pop`: Returns the top integer at `stack.arr[stack.n-1]` and decrements `stack.n` to update how many spots are being used. **If the stack is empty, returns -1.**
- `printStack`: Prints all the items currently on the stack by going through the first `stack.n` elements in `stack.arr`.

Do not modify the `main` function. Once the above functions are correctly filled in, the program should run as follows:

```
Making stack...
Test 1: Single push+pop
Pushed 5 onto the stack.
Popped 5 off the stack.

Test 2: Fill stack
Successfully pushed 0!
Successfully pushed 1!
Successfully pushed 2!
Successfully pushed 3!
Successfully pushed 4!
Stack full!

Test 3: Empty stack
Successfully popped 4!
Successfully popped 3!
Successfully popped 2!
Successfully popped 1!
Successfully popped 0!
Stack empty!
```

Figure 2. Correct output for `Lab10b.c`.

Hint: Note that `push` and `pop` accept a pointer to the stack so that it can be modified in-place. Recall that the three accesses below can all be used to get to a given field of some `struct str`:

`str.field` `(*strPointer).field` `strPointer->field`

Submission details:

- Upload a compressed archive (e.g., .zip) containing `Lab10a.c and Lab10b.c`.
- The archive should be named `Lab10_LastName`, where `LastName` is your last name.

- If you're on Linux, you can use the following command to create a .tar.gz archive from the terminal:

  ```
  $ tar -czvf Lab10_LastName.tar.gz Lab10a.c Lab10b.c
  ```

  where `LastName` is your last name.