CS2123 Data Structures Summer 2017

Assignment 2: Stacks and Recursive Programs

Due 6/26/17 by 11:59pm

1. Stacks in C (15 points)

Implement a stack to store and remove strings that will be read from a file. The data file contains a series of strings, one per line. Each string will contain 255 or fewer characters. Whenever you read the string "pop", this is a signal to pop your stack. Any other string should be pushed onto your stack.

The format of the data file is:
<string to push>
<string to push>
⋮
pop
<string to push>
pop
etc.

To create the initial stack, use malloc to allocate enough space to store 10 strings. Keep track of how many elements are in your stack. When you stack reaches capacity, your push method needs to allocate more space to your stack before pushing the next element (add space for another 10 strings). You can use realloc, or something else like malloc/copy/swap. You do not ever need to shrink your stacks capacity.

You are required to implement the following stack functions: push, pop, empty, and full.

- **create** returns a new empty stack.
- **push** takes a string parameter which is the value it pushes onto the stack. It may also need to call realloc to expand the size of the stack before completing the push.
- **pop** returns the string that was removed from the stack.
- **empty** returns TRUE if the stack has no elements, otherwise FALSE.
- **full** returns TRUE if the stack does not have any room left, otherwise FALSE.

Your program must print the assignment 2 and your name. Additionally, each time you read "pop" (i.e., each time you receive a signal to pop the stack) you should print the # of elements in the stack after popping and also print the string that is popped off the stack. You should also print a message every time your stack grows. For example, the program might print the following:

```
Assignment 2 Problem 1 by <your name>
# elements after popping: 2          string popped: Be
# elements after popping: 1          string popped: sure
# elements after popping: 0          string popped: to
Stack capacity has grown from 10 elements to 20 elements
```

Continued on the back ↪

2. Recursion in C (15 points)

The greatest common divisor (GCD) of two numbers is the largest number which divides both of them (e.g., $GCD(20, 15) = 5$, $GCD(20, 20) = 20$, $GCD(20, 9) = 1$, ...). We define the GCD as follows:

$$\begin{array}{ll} GCD(x, y) = y & \text{if}(y <= x \text{ and } x\%y == 0) \\ GCD(x, y) = GCD(y, x) & \text{if}(x < y) \\ GCD(x, y) = GCD(y, x\%y) & \text{otherwise} \end{array}$$

Use this definition to create a recursive algorithm for GCD.

In addition to this you should create an iterative algorithm for GCD. Since $1 \leq GCD(x, y) \leq minimum(x, y)$, you can test all of the values in this range using a while loop and return the largest one.

Clearly, the results from both the recursive and iterative functions should be the same otherwise something is wrong with one or both of your functions.

The main which will test your functions has been provided.

Deliverables:

Answers to homework problems should be submitted as two C source code files, one file for each problem. Archive and submit the files in Blackboard under Assignment 2.

Remember:

The program you submit should be the work of only you. Cheating will be reported to judicial affairs. Both the copier and copiee will be held responsible.