Exercise number 3 (due 10 Jun):

Exercise 4.3 (page 93)**:** Rewrite reverse to use an array pointer instead of a slice.

Exercise 4.4: Write a version of rotate that operates in a single pass.

Exercise 4.5: Write an in-place function to eliminate adjacent duplicates in a []string slice.

Exercise 4.6: Write an in-place function that squashes each run of adjacent Unicode spaces (see unicode.IsSpace) in a UTF-8-encoded []byte slice into a single ASCII space.

Exercise 4.7: Modify reverse to reverse the characters of a []byte slice that represents a UTF-8-encoded string, in place. Can you do it without allocating new memory?

Exercise 5.10 (page 140)**:** Rewrite topoSort to use maps instead of slices and eliminate the initial sort. Verify that the results, though nondeterministic, are valid topological orderings.

Exercise 5.11: The instructor of the linear algebra course decides that calculus is now a prerequisite. Extend the topoSort function to report cycles.

Exercise 5.12: The startElement and endElement functions in gopl.io/ch5/outline2 (§5.5) share a global variable, depth. Turn them into anonymous functions that share a variable local to the outline function.

Exercise 5.13: Modify crawl to make local copies of the pages it finds, creating directories as necessary. Don’t make copies of pages that come from a different domain. For example, if the original page comes from golang.org, save all files from there, but exclude ones from vimeo.com.

Exercise 5.14: Use the breadthFirst function to explore a different structure. For example, you could use the course dependencies from the topoSort example (a directed graph), the file system hierarchy on your computer (a tree), or a list of bus or subway routes downloaded from your city government’s web site (an undirected graph).

Exercise 3.1 (page 60)**:** If the function f returns a non-finite float64 value, the SVG file will contain invalid <polygon> elements (although many SVG renderers handle this gracefully). Modify the program to skip invalid polygons.

Exercise 3.2: Experiment with visualizations of other functions from the math package. Can you produce an egg box, moguls, or a saddle?

Exercise 3.3: Color each polygon based on its height, so that the peaks are colored red (#ff0000) and the valleys blue (#0000ff).

Exercise 3.4: Following the approach of the Lissajous example in Section 1.7, construct a web server that computes sur faces and writes SVG data to the client. The server must set the ContentType header like this:

w.Header().Set("ContentType", "image/svg+xml")

(This step was not required in the Lissajous example because the server uses standard heuristics to recognize common formats like PNG from the first 512 bytes of the response and generates the proper header.) Allow the client to specify values like height, width, and color as HTTP request parameters.