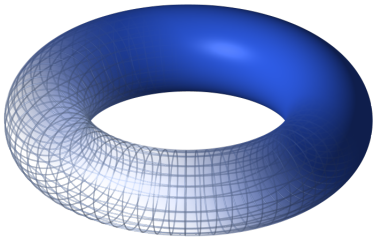
CS1132 Spring 2014 Assignment 1a due 2/12 11:59pm

> *Adhere to the Code of Academic Integrity.* You may discuss background issues and general strategies with others and seek help from course staff, but the implementations that you submit must be your own. In particular, you may discuss general ideas with others but you may not work out the detailed solutions with others. It is never OK for you to see or hear another student's code and it is never OK to copy code from published/Internet sources. If you feel that you cannot complete the assignment on you own, seek help from the course staff.

When submitting your assignment, follow the instructions summarized in Section 4 of this document.

Do not use the `break` or `continue` statement in any homework or test in CS1132.

1 Volume of Torus



In geometry, a torus is a surface of revolution generated by revolving a circle in three-dimensional space about an axis coplanar with the circle. If the axis of revolution does not touch the circle, the surface has a ring shape and is called a ring torus or simply torus if the ring shape is implicit. [1] The surface area and interior volume of a torus are computed using Pappus's centroid theorem giving,

$$A = 4\pi^2 Rr$$
$$V = 2\pi^2 Rr^2$$

where R is the distance from the center of the tube to the center of the torus, and r is the radius of the tube. In addition, the following constrain must satisfy: $R > 0$ and $r > 0$. If either of the input arguments is out of the range, the output should be 0 for A and V. The file *torus.m* defines a function with R and r as input arguments, and output A the area and V the volume as output.

- Implement the following function:

```
function [A, V] = torus(R,r)
% This function computes the surface area and interior volume of a torus
% Input (R,r):
% R: the distance from the center of the tube to the center of the torus
% r: the radius of the tube.
% Output (A,V): A: the area; V: the volume
```

[1] Cited from Wikipedia

2 Bubble Sort

Bubble sort is a very simple sorting algorithm: it works by repeatedly going through the array to be sorted, comparing two adjacent items at a time and swapping them if they are in the wrong order. For example, in a pass through the array item 1 and 2 are compared and possibly swapped, then items 2 and 3 are compared and possibly swapped, and so forth until the end of the array is reached. The algorithm passes through the array as long as swaps continue to be needed, that is, the array is not yet completely sorted. The algorithm gets its name from the way smaller elements "bubble" to the beginning of the array.
Implement this algorithm in the following function:

```
function sortedArray = bubbleSort(A)
% This function implements bubble sort.
% Input A: A can be a vector or matrix which stores the number that needs to be
% sorted. If A is a matrix, convert it to a vector first.
% Output: sortedArray is the sorted array after the bubble sort
```

Note that A could be a vector or a matrix. When implementing this function, one possible approach is to convert A to a vector first.

3 Three Dice

(i) Random integer generator

Using the `help` command, learn about the `rand, ceil, fix`, and `floor` functions. After reading their brief description, implement a random integer generator using those functions only. Implement the following function:

```
function result = myRandInt(startInt,endInt)
% Returns an integer from startInt to endInt, inclusive with equal probability.
% If startInt > endInt, treat startInt as the upper bound and endInt as the lower
% bound instead.
```

(ii) Constrained integer generator

We would like to randomly generate a pair of numbers (i, j). Integer i is in between $startInt$ to $endInt$. Integer j satisfies $i \leq j \leq 2i$. Make effective use of function `myRandInt` from the previous problem.
Implement the following function:

```
function [i,j] = constrainedPairIntegers(startInt,endInt)
% Generate a pair of integer (i,j),
% Input: startInt: start index, endInt: end index.
% Output i's range is (startInt, endInt) j is in the range of i<=j<=2i
```

4 Self-check list

The following is a list of the minimum *necessary* criteria that your assignment must meet in order to be considered *satisfactory*. Failure to satisfy any of these conditions will result in an immediate request to resubmit your assignment. Save yourself and the graders time and effort by going over it before submitting your assignment for the first time. Although all these criteria are necessary,

meeting all of them might still not be *sufficient* to consider your submission satisfactory. We cannot list everything that could be possibly wrong with any particular assignment!

- Δ Comment your code! Make sure your functions are properly commented, regarding function purpose and input/output parameters.
- Δ Suppress all unnecessary output by placing semicolons (;) appropriately. At the same time, make sure that all output that your program intentionally produces is formatted in a user-friendly way.
- Δ Make sure your functions names are *exactly* the ones we have specified, *including* case.
- Δ Check that the number and order of input and output parameters for each of the functions match exactly the specifications we have given.
- Δ Test each one of your functions independently, whenever possible, or write short scripts to test them.
- Δ Check that your scripts do not crash (i.e., end unexpectedly with an error message) or run into infinite loops. Check your script several times in a row. Before each test run, type the commands `clear all; close all;` to delete all variables in the workspace and close all figure windows.

5 Submission instructions

1. Upload files `torus.m`, `bubbleSort.m`,`myRandInt.m`, `constrainedPairIntegers.m`, to CMS in the submission area corresponding to Assignment 1a in CMS before the deadline. Late submission is accepted up to 24 hours after the deadline with a 10% penalty.
2. *After grading:* If you resubmit the assignment, upload your corrected files and *be sure to select the* ***Regrade Request*** *option.*