Name: ______________________________ NetID: __________
(Legibly print last name, first name, middle name)

Statement of integrity: *I did not, and will not, violate the rules of academic integrity on this exam.* ______________________________
(Signature)

Instructions:

- This is an open-note exam and is designed to take approximately 50 minutes to complete. No collaboration is allowed.
- The exam is worth a total of 50 points.
- Read each problem completely before starting it.
- If a question is unclear, e-mail Dr. Muhlberger; do not ask anyone else. Check Canvas before submitting your exam in case we need to announce any clarifications to the whole class.
- Clarity, conciseness, and good programming style count for credit.
- Indicate your final answer. If you supply multiple answers, you may receive a *zero*.
- Use only MATLAB code. No credit for code written in other programming languages.
- Assume there will be no input errors.
- Vectorized code is not required (but it may make things easier to write).
- You may write a subfunction as part of your solution if you think it will clarify your code.
- Do not use `switch`, `try`, `catch`, `break`, `continue`, or `return` statements.
- **Do not use built-in functions that have not been discussed in the course.** Limit yourself to the following MATLAB predefined functions: `rem`, `floor`, `ceil`, `round`, `min`, `max`, `sum`, `rand`, `zeros`, `ones`, `linspace`, `length`, `size`, `strcmp`, `str2double`, `cell`, `sort`, `fopen`, `fgetl`, `feof`, `fclose`, `input`, `fprintf`, `disp`

Examples:
- `rem(5,2)` → 1, the remainder of 5 divided by 2
- `min(-4,3)` → -4, smallest argument
- `max(-4,3)` → 3, largest argument
- `sum([0 4; 1 -1])` → [1 3], vector of column sums of the *matrix* argument
- `rand()` → a random real value in the interval (0,1)
- `zeros(1,4)` → 1 row 4 columns of zeros
- `length([2 4 8])` → 3, length of a vector
- `[nr,nc,np]=size(M)` → dimensions of `M`: `nr` rows, `nc` columns, `np` layers
- `strcmp('cat','dog')` → 0, the two strings are not identical
- `str2double(' 3.14')` → 3.14, character vector interpreted as a number
- `[y,idx]=sort(x)` → elements of `x` sorted in ascending order returned in `y` with the property that `y(k)=x(idx(k))`
- `fid=fopen('file.txt', 'r')`: Open the file `file.txt` for reading
- `line=fgetl(fid)`: Read the next line from an open file
- `done=feof(fid)`: `done` is 1 if there is nothing left to read in file
- `fclose(fid)`: Close a previously opened file

Question 1. (18 points)

When gym teacher Jen needs to divide her class into teams, she first asks them to line up from shortest to tallest. Then, starting with the shortest student, she has them count off "one," "two," "three," etc., starting over with "one" after someone says N, where N is the number of teams. Then all of the "ones" team up, all of the "twos" team up, etc.

Implement the following function to simulate this procedure in MATLAB, forming three teams. The input and output are cell arrays containing handles to objects of class `Student`; a `Student` has two public properties: `name` (a character vector representing their name) and `height` (a double representing their height in inches). The students in the input array are unsorted, and the number of students might not be a multiple of 3.

```
function teams = teamUp(students)
% Divide students into three teams according to their height.
% Given `students`, a 1D cell array of handles to Student objects,
% sort the students by height and assign them to the 2D cell array
% `teams` where the first column contains students on team "one", the
% second column contains students on team "two", and the third column
% contains students on team "three".  Students are assigned such that
% the shortest, and every third student thereafter, is on team "one",
% the students one taller than each of them are on team "two", and the
% remaining students are on team "three".  Within each column, the
% rows contain the students on that team in ascending order of height.
```

Question 2. (18 points)

The file Statement-Apr.txt contains a customer's latest bank statement: a list of withdrawals and deposits that occurred over the past month. An example file is shown below:

```
ID             4710
ACTIVITY    DATE         AMOUNT
DEPOSIT     4/1/20       100.00
DEPOSIT     4/15/20      100.00
WITHDRAW    4/16/20       15.26
DEPOSIT     4/23/20       55.00
WITHDRAW    4/30/20       79.17
```

The ID and DATE values are in character columns 13–19, and the AMOUNT values are in character columns 25–30.

A second file, Balances-Mar.txt, contains all customers' balances at the end of the previous month. Each line corresponds to one customer (whose **customer ID** matches the line number), and the contents of the line are a decimal number indicating the number of dollars in the customer's account (see example below; a line may be any length, but the only other characters will be spaces). For the above example, the customer's initial balance would be found on the 4710th line in the file. As another example, given the balances file below,

```
 517.46
 213.59
1337.88
 117.24
  99.99
```
⋮ *(many more lines)*

the starting balance for the statement with customer ID "4" will be $117.24.

Implement the following function to compute the customer's final balance at the end of the statement. For each DEPOSIT line, the amount should be added to their running balance, while for each WITHDRAW line, the amount should be subtracted from their balance. Additionally, the function should determine the minimum balance that was in the account at some time over the course of the month (may be negative). You may assume that the statement file will have an ID on the first line (followed by a header on the second line) and that the balances file will have at least as many lines as the ID value. When reading the statement, lines other than DEPOSIT and WITHDRAW (after the ID) should be ignored.

(Write your answer on the following page. If you are using your own paper, be sure to scan a placeholder sheet for page 3.)

```
function [fbal, minbal] = updateBalance()
% Compute a customer's balance based on their monthly statement.
% The customer's statement is read from the file Statement-Apr.txt,
% and the starting balances for all customers are read from the file
% Balances-Mar.txt.  `fbal` will contain the customer's final balance
% at the end of the statement, and `minbal` will contain the smallest
% balance in the account after any transaction over the course of the
% month.
```

Question 3. (14 points)

Consider a class named `Vert` (inheriting from `handle`) with public properties `x`, `y`, and `z`. `Vert`'s constructor takes 3 arguments: the initial values of its propertiets in x, y, z order. `Vert` also has three public methods:

- `scale(self, s)` multiplies all of the instance's properties by the value of `s`
- `flip(self)` copies the instance's value of `x` to its property `y`, the old value of `y` to `z`, and the old value of `z` to `x`
- `v = add(self, other)` returns a new instance of `Vert` whose properties are the sum of the corresponding properties in `self` and `other` (neither of which is modified)

(a) Declare and implement the `scale()` method as described above. Assume the code you write will go in a `methods` section. You do not need to write a header comment.

(b) If the following code were executed, what output would be produced?

```
a = Vert(1, 2, 5);
b = a;
b.x = 3;
disp(a.x)
a.scale(2)
disp(b.z)
a = Vert(5, 12, 13);
a.flip();
disp(b.y)
b = Vert(3, 1, 4);
c = b.add(a);
disp(b.z)
disp(c.x)
```

Output: