```
% genNvectors.m
% Patrick Utz, 3/23/18, 10.1

% Problem: Write a MATLAB program genNvectors.m to prompt the user
% to input a positive integer N, and generate N random vectors, each
% with 10 random numbers uniformly distributed between 0 and 1. Each
% vector will be stored in a variable with a name called ?vector1?,
% ?vector2?, ?vector3?..., ?vectorN?. Your program should perform error
%     checking for the user input value of N. After testing your program,
% there should be N variables, vector1,..., vectorN. created in your
%     workspace, each with 10 random numbers.

% Variables: N = positive integer inputted by user, expres = expression
% to be evaluated to create the individual variables

clear
N = input('Hello! Please enter a positive integer N: ');
while N < 0 || int32(N) ~= N
N = input('Hello! Please enter a positive integer N: ');
end

for k = 1:N
   expres = sprintf('vector%d = (1-0)*rand(1,10)+0',k);
   eval(expres);
end

>> genNvectors
Hello! Please enter a positive integer N: 3

vector1 =

   0.1403    0.2601    0.0868    0.4294    0.2573    0.2976    0.4249    0.1192    0.4951    0.7064


vector2 =

   0.2436    0.7851    0.0741    0.3939    0.0034    0.2207    0.0013    0.1892    0.1425    0.2681


vector3 =

   0.1749    0.1386    0.5989    0.9011    0.9394    0.2212    0.4827    0.3760    0.5238    0.2649
```

```
% wordparser.m
% Patrick Utz, 3/23/18, 10.2

% Problem: Write a MATLAB function called wordparser that can parse
% words for a text document. The function should take the document as
% input, and return one single output argument which stores all distinctive
% words (not case sensitive) in the document. Your program should be able
% to eliminate any punctuations leading or trailing each word. It should
% also be able to handle an empty input. Use the following input to test
% your program:
% 1. This is a test.
% 2. Good!
% 3. He said, ?Go, go for it!?

% Variables: words = used to modify and output the cell array with only
% words

function words = wordparser(text)
% wordparser seperates text into individual words
% Format of call: wordparser( input text )
% Returns cell array with individual words

words = text;
for k = 1:length(words)
   if ~isletter(words(k))
      words(k) = ' ';
   end
end
words = deblank(words);
words = strsplit(words,' ')
```

```
% wordparserMain.m
% Patrick Utz, 3/23/18, 10.2

clear
wordparser('This is a test.');
```

```
>> wordparserMain
words =
  1×4 cell array
    {'This'}    {'is'}    {'a'}    {'test'}
```

```
>> wordparserMain
words =
  1×1 cell array
    {'Good'}

>> wordparserMain
words =
  1×6 cell array
    {'He'}    {'said'}    {'Go'}    {'go'}    {'for'}    {'it'}
```

```
% StudentInfoMain.m
% Patrick Utz, 3/23/18, 10.3

% Problem: Write a MATLAB program that will call the following two
% functions.
% 1) EnterStudentInfo.m: this function takes a positive integer N as the
% input arguments, and return a single output argument, which is a cell
% array that stores the names, student IDs and GPAs for a total of N
% students. The function should prompt the user to input appropriate
% information interactively. An example is illustrated below.
% >> students = EnterStudentInfo(2)
% Please enter the name of student No. 1: John
% Please enter the student ID for John: 1234
% Please enter the GPA for John: 4.0
% Please enter the name of student No. 2: Mary
% Please enter the student ID for Mary: 5678
% Please enter the GPA for Mary: 5.0
% The result of the above testing should result in the following
% variable: >> students
% students =
% 2×3 cell array
% 'John' [1234] [4] 'Mary' [5678] [5]
% 2) PrintStudentInfo.m: this function takes one single input argument
%     with all the above mentioned student information, and print in a
%     table format with a heading line as shown below.
% >> PrintStudentInfo(students)
% Student Name John
% Mary
% Student ID GPA 1234 4.00 5678 5.00

% Variables: students = student cell array

clear

students = EnterStudentInfo(2);
PrintStudentInfo( students );


% EnterStudentInfo.m
% Patrick Utz, 3/23/18, 10.3

function students = EnterStudentInfo(N)
% EnterStudentInfo takes a positive integer N as the input arguments,
% and returns a single output argument, which is a cell array that stores
% the names, student IDs and GPAs for a total of N students. The function
% prompts the user to input appropriate information interactively.
% Format of call: EnterStudentInfo( positive N integer of # students )
% Returns the cell array
```

```
students = cell(N,3);
for k = 1:N
   namePrompt = sprintf('Please enter the name of student No. %d: ',k);
   students(k,1) = { input(namePrompt, 's') };
   ID_Prompt = sprintf('Please enter the student ID for %s: ', students{k,1});
   students(k,2) = { input(ID_Prompt) };
   GPA_Prompt = sprintf('Please enter the GPA for %s: ', students{k,1});
   students(k,3) = { input(GPA_Prompt) };
end
```

```
% PrintStudentInfo.m
% Patrick Utz, 3/23/18, 10.3

function PrintStudentInfo(students)
% PrintStudentInfo takes one single input argument of the student
% information, and prints it all in a table format
% Format of call: PrintStudentInfo( cell array of student info )
% Returns student info in table format

fprintf('Student Name \t Student ID \t GPA\n')
[row,columns] = size(students);
for k = 1:row
fprintf('\t%s\t\t%d \t %#1.2f \n', students{k,1}, students{k,2}, students{k,3})
end
```

```
>> StudentInfoMain
Please enter the name of student No. 1: John
Please enter the student ID for John: 1234
Please enter the GPA for John: 4.0
Please enter the name of student No. 2: Mary
Please enter the student ID for Mary: 5678
Please enter the GPA for Mary: 5.0
Student Name      Student ID        GPA
          John              1234      4.00
          Mary              5678      5.00
```

% SubjectsMain.m
% Patrick Utz, 3/23/18, 10.4

% Problem: Create and run a MATLAB program to solve exercise 20 of Chapter
% 8 with the following changes: The main program should call three
% functions:
% 1) datagen.m: generate and return (as output arguments) a structure
% array of N subjects, where N is the input argument. The names of the
% subjects can be generated from the 26 alphabets randomly with 3 ? 8
% letters (all in lower case). The id is simply the index of the subject
% in the structure array. The heights of the subjects is uniformly
% distributed random numbers between 5.5 and 6.8; The weights of the
% subjects is uniformly distributed between 90 and 250;
% 2) calcavg.m: calculate the average height and weight of all subject
% and return them as output arguments.
% 3) printdata.m: print the generated subjects and the results to the
% screen as shown below

```
 Name          Id            Height         Weight
 joey          1             6.7            222.2
 mary          2             5.6            112.5
    …             …             …              …
Average height is 6.3
Average weight is 125.7
Eligible subjects are:  mary
     ....
```

20. A script stores information on potential subjects for an experiment in a vector of structures called *subjects*. The following shows an example of what the contents might be:

```
>> subjects(1)
ans =
      name: 'Joey'
    sub_id: 111
    height: 6.7000
    weight: 222.2000
```

CHAPTER 8: Data Structures

For this particular experiment, the only subjects who are eligible are those whose height or weight is lower than the average height or weight of all subjects. The script will print the names of those who are eligible. Create a vector with sample data in a script, and then write the code to accomplish this. Don't assume that the length of the vector is known; the code should be general.

% Variables: subjects = structure of subjects, avgHeight = average
% height of subjects, avgWeight = average weight of subjects

```
clear

subjects = datagen(4);
[avgHeight, avgWeight] = calcavg(subjects);
printdata(subjects, avgHeight, avgWeight);
```

% datagen.m
% Patrick Utz, 3/23/18, 10.4

```
function subjects = datagen(N)
```
% datagen generates and returns (as output arguments) a structure array
% of N subjects, where N is the input argument. The names of the subjects

```
% can be generated from the 26 alphabets randomly with 3 ? 8 letters (all
% in lower case). The id is simply the index of the subject in the
% structure array. The heights of the subjects is uniformly distributed
% random numbers between 5.5 and 6.8; The weights of the subjects is
% uniformly distributed between 90 and 250
% Format of call: datagen( N number of subjects )
% Returns structure array of N subjects

alphabet = ['a' 'b' 'c' 'd' 'e' 'f' 'g' 'h' 'i' 'j' 'k' 'l' 'm' 'n'...
   'o' 'p' 'q' 'r' 's' 't' 'u' 'v' 'w' 'x' 'y' 'z'];
for k = 1:N
   name = alphabet(randi([1,26],1,randi([3,8])));
   height = (6.8-5.5)*rand+5.5;
   weight = (250-90)*rand+90;
   subjects(k) = struct('Name',name,'Id',k,'Height',height,'Weight',weight)
end
end
```

```
% calcavg.m
% Patrick Utz, 3/23/18, 10.4

function [avgHeight, avgWeight] = calcavg(subjects)
% calcavg calculates the average height and weight of all subjects and
% returns them as output arguments
% Format of call: calcavg( structure array of subjects )
% Returns average height and weight of all subjects

avgHeight = sum([subjects.Height])/length(subjects);
avgWeight = sum([subjects.Weight])/length(subjects);
end
```

```
% printdata.m
% Patrick Utz, 3/23/18, 10.4

function printdata(subjects, avgHeight, avgWeight)
% printdata prints the generated subjects and the results to the screen
% Format of call: printdata()
% Returns cell array with individual words

fprintf('Name \t\t\tID \t\t Height \t\tWeight \n')
qualified = {};
for k = 1:length(subjects)
   if subjects(k).Height < avgHeight || subjects(k).Weight < avgWeight
      qualified(length(qualified)+1) = {subjects(k).Name};
   end
   fprintf('%s \t\t\t%d \t\t %#1.1f \t\t\t% #1.1f \n', subjects(k).Name, subjects(k).Id, subjects(k).Height,
subjects(k).Weight);
```

```
end
fprintf('Average height is %#1.1f\nAverage weight is %#1.1f\nEligible subjects are: %s',avgHeight,avgWeight,'')
disp(qualified)
end
```

subjects =

Name	ID	Height	Weight
yde	1	6.4	198.0
cajul	2	5.6	218.9
lnnqqe	3	6.5	121.5
ghayn	4	6.1	173.8

```
Average height is 6.2
Average weight is 178.0
Eligible subjects are:     'cajul'    'lnnqqe'    'ghayn'
```

% QuizOrganizer.m
% Patrick Utz, 3/23/18, 10.5

% Problem: Create and run a MATLAB program to solve exercise 21 of
% Chapter 8. Write a function to create the data file named studentinfo.dat.
% You can use the examples in the book as the contents in the data file,
% or create your own one.
% Variables: row = length of row of structure, column = length of column
% of structure, studentinfo matrix with raw info, studentInfoVec =
% structure of student info

21. Quiz data for a class is stored in a file. Each line in the file has the student ID number (which is an integer) followed by the quiz scores for that student. For example, if there are four students and three quizzes for each, the file might look like this:

```
44    7     7.5     8
33    5.5   6       6.5
37    8     8       8
24    6     7       8
```

First create the data file, and then store the data in a script in a vector of structures. Each element in the vector will be a structure that has 2 members: the integer student ID number and a vector of quiz scores. The structure will look like this:

		students		
	id_no	quiz 1	2	3
1	44	7	7.5	8
2	33	5.5	6	6.5
3	37	8	8	8
4	24	6	7	8

To accomplish this, first use the **load** function to read all information from the file into a matrix. Then, using nested loops, copy the data into a vector of structures as specified. Then, the script will calculate and print the quiz average for each student.

```
clear

CreateStudentInfo();
load studentinfo.dat;
[row, column] = size(studentinfo);
for k = 1:row
   studentinfo(k,1) = int32(studentinfo(k,1));
end

for i = 1:row
   studentInfoVec(i) = struct('id_no',studentinfo(i,1),'quiz',studentinfo(i,2:end));
end

for j = 1:row
   fprintf('Student %d average: %f\n', studentInfoVec(j).id_no,(sum([studentInfoVec(j).quiz]))/(column-1));
end
```

% calcavg.m
% Patrick Utz, 3/23/18, 10.5

```
function CreateStudentInfo()
```
% CreateStudentInfo creates a data file named studentinfo.dat
% Format of call: CreateStudentInfo( )
% Returns nothing but creates a data file named studentinfo.dat

```
studentinfo = [44 7 7.5 8; 33 5.5 6 6.5; 37 8 8 8; 24 6 7 8];
save studentinfo.dat studentinfo -ascii;
end
```

```
>> QuizOrganizer
Student 44 average: 7.500000
Student 33 average: 6.000000
Student 37 average: 8.000000
Student 24 average: 7.000000
```