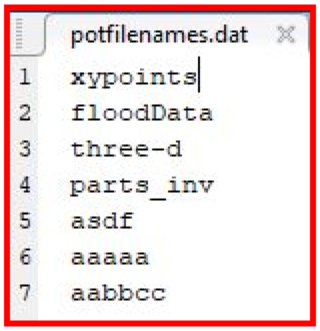
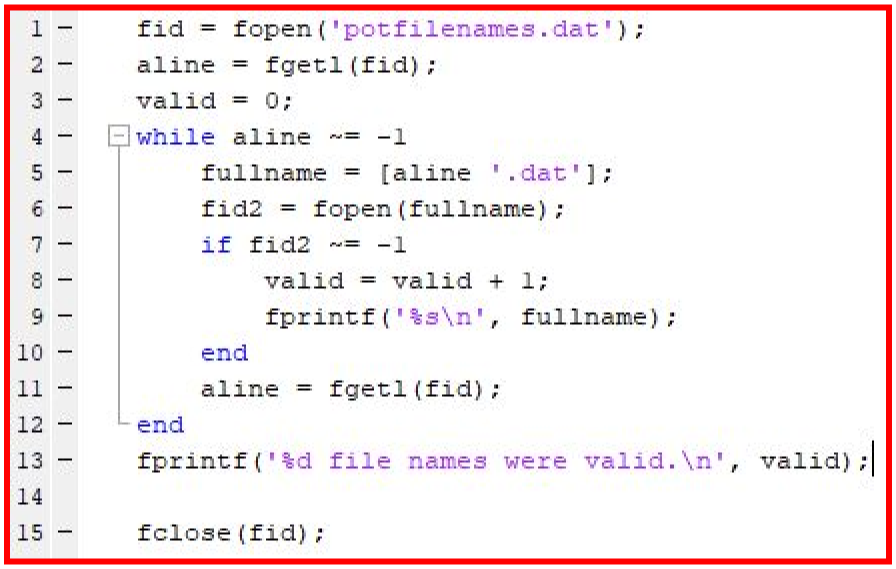
ENG EK 125 - Worksheet 9

Name: Ande Chen Section: C1

1) A file "potfilenames.dat" stores potential file names, one per line. The names do not have any extension. Write a script that will print the names of the valid files, once the extension ".dat" has been added. "Valid" means that the file exists in the Current Directory, so it could be opened for reading. The script will also print how many of the file names were valid.

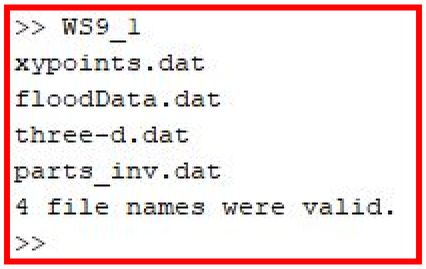


```
xypoints
floodData
three-d
parts_inv
asdf
aaaaa
aabbcc
```



```
fid = fopen('potfilenames.dat');
aline = fgetl(fid);
valid = 0;
while aline ~= -1
    fullname = [aline '.dat'];
    fid2 = fopen(fullname);
    if fid2 ~= -1
        valid = valid + 1;
        fprintf('%s\n', fullname);
    end
    aline = fgetl(fid);
end
fprintf('%d file names were valid.\n', valid);

fclose(fid);
```

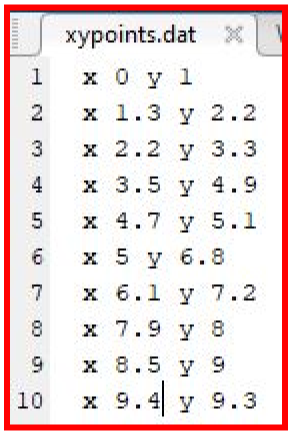


```
>> WS9_1
xypoints.dat
floodData.dat
three-d.dat
parts_inv.dat
4 file names were valid.
>>
```

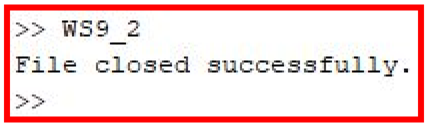
2) Write a script that will read from a file x and y data points in the following format:

```
x 0 y 1
x 1.3 y 2.2
```

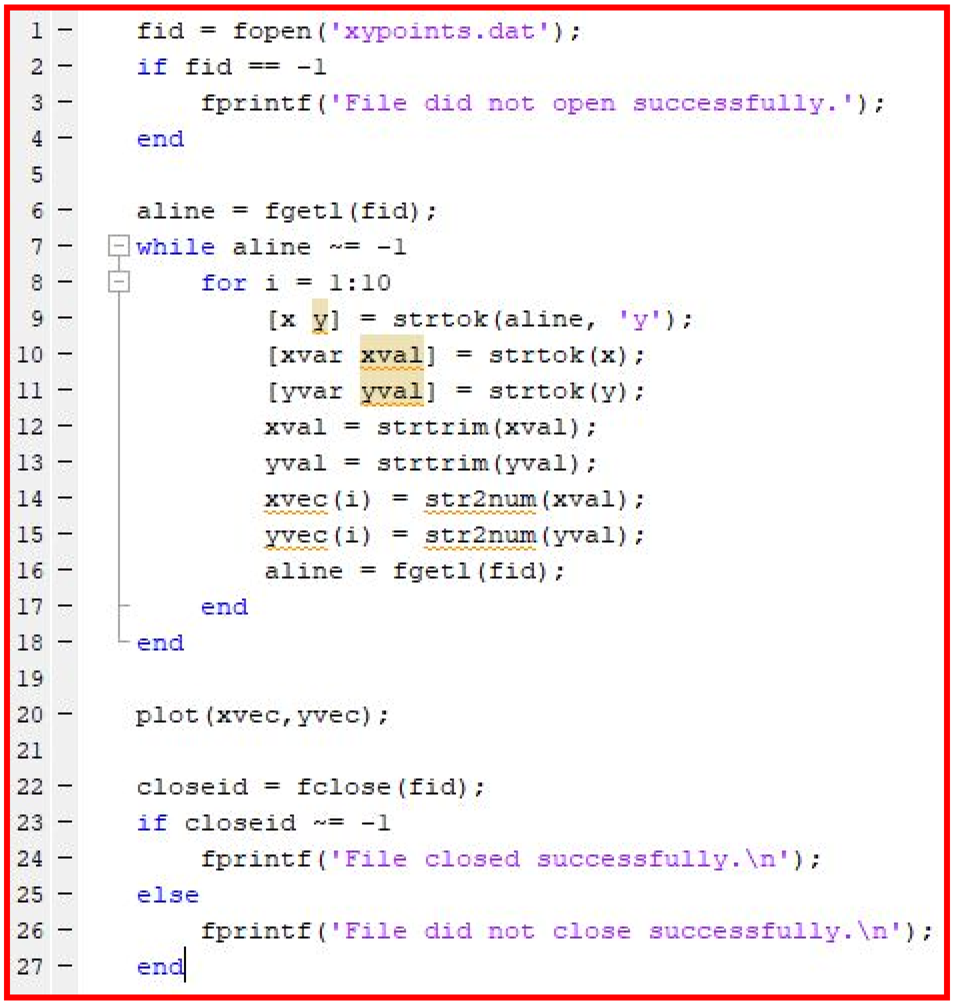
The format of every line in the file is the letter 'x', a space, the x value, space, the letter 'y', space, and the y value. First, create the data file with 10 lines in this format. Do this by using the Editor/Debugger, then File Save As *xypoints.dat*. The script will attempt to open the data file and error-check to make sure it was opened. If so, it uses a **<u>for</u>** loop and **fgetl** to read each line as a string. In the loop, it creates x and y vectors for the data points. After the loop, it plots these points and attempts to close the file. The script should print whether or not the file was successfully closed.



```
x 0 y 1
x 1.3 y 2.2
x 2.2 y 3.3
x 3.5 y 4.9
x 4.7 y 5.1
x 5 y 6.8
x 6.1 y 7.2
x 7.9 y 8
x 8.5 y 9
x 9.4 y 9.3
```



```
>> WS9_2
File closed successfully.
>>
```



```
fid = fopen('xypoints.dat');
if fid == -1
    fprintf('File did not open successfully.');
end

aline = fgetl(fid);
while aline ~= -1
    for i = 1:10
        [x y] = strtok(aline, 'y');
        [xvar xval] = strtok(x);
        [yvar yval] = strtok(y);
        xval = strtrim(xval);
        yval = strtrim(yval);
        xvec(i) = str2num(xval);
        yvec(i) = str2num(yval);
        aline = fgetl(fid);
    end
end

plot(xvec,yvec);

closeid = fclose(fid);
if closeid ~= -1
    fprintf('File closed successfully.\n');
else
    fprintf('File did not close successfully.\n');
end
```

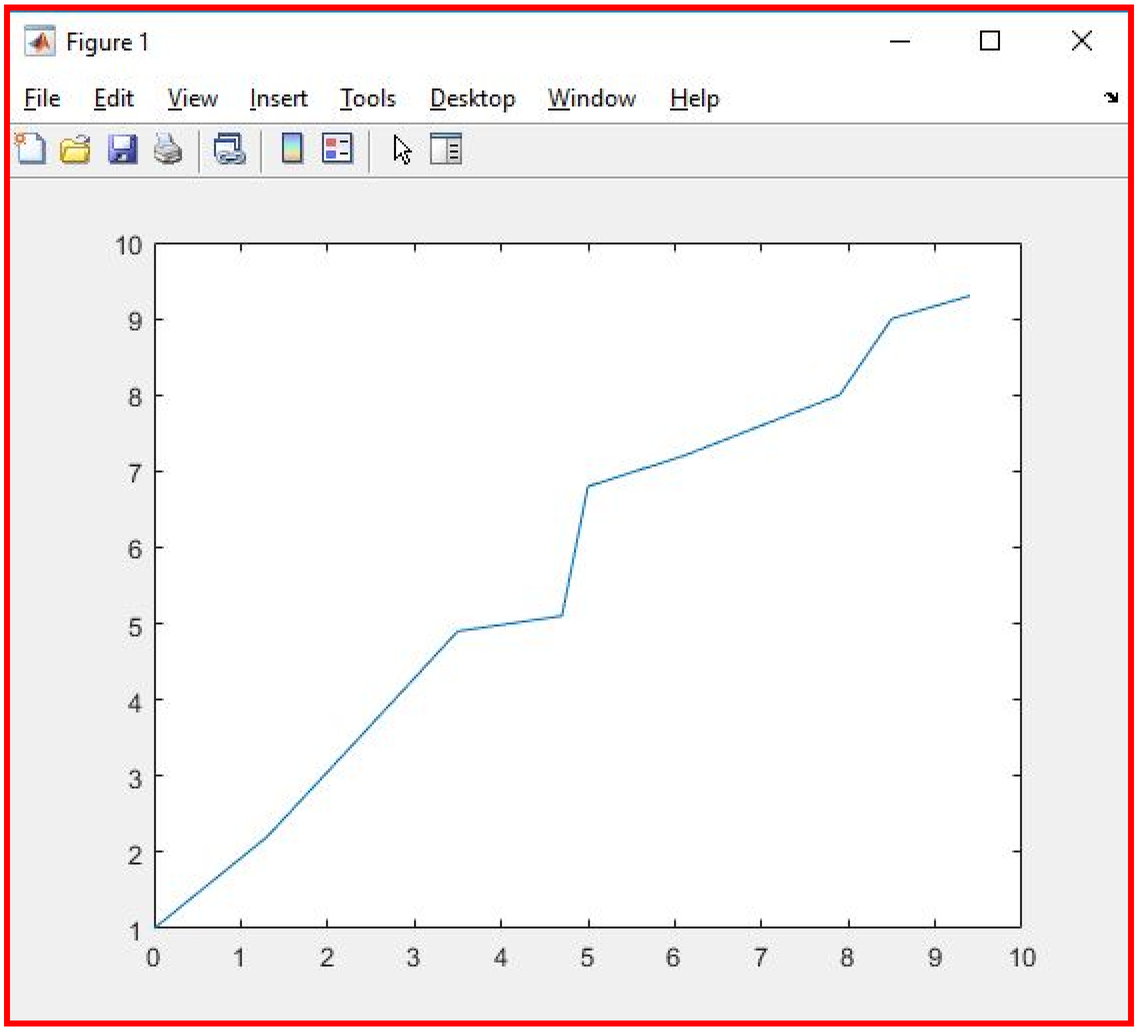
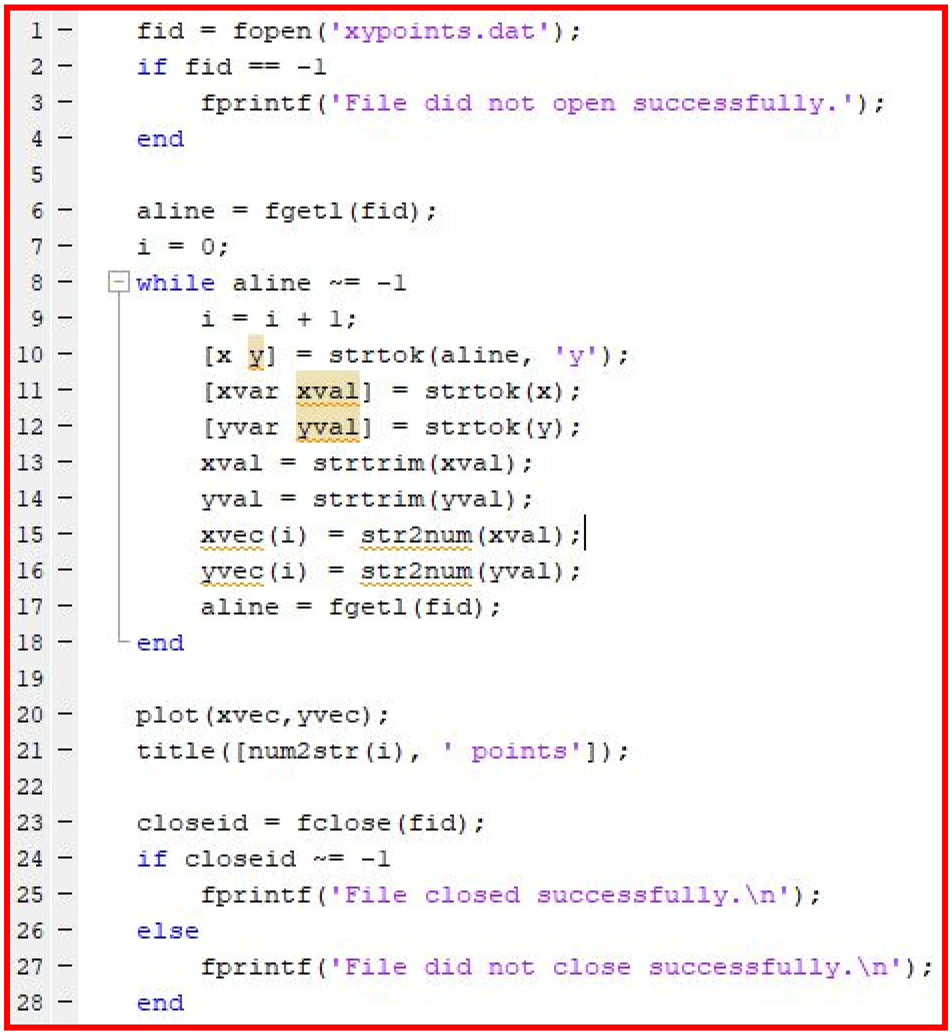


Figure 1
File
Edit
View
Insert
Tools
Desktop
Window
Help
0
1
2
3
4
5
6
7
8
9
10
1
2
3
4
5
6
7
8
9
10

3) Modify the script from the previous problem. Assume that the data file is in exactly that format, but do not assume that the number of lines in the file is known. Instead of using a **for** loop, loop until the end of the file is reached. The number of points, however, should be in the plot title.

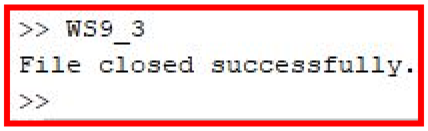


```
fid = fopen('xypoints.dat');
if fid == -1
    fprintf('File did not open successfully.');
end

aline = fgetl(fid);
i = 0;
while aline ~= -1
    i = i + 1;
    [x y] = strtok(aline, 'y');
    [xvar xval] = strtok(x);
    [yvar yval] = strtok(y);
    xval = strtrim(xval);
    yval = strtrim(yval);
    xvec(i) = str2num(xval);
    yvec(i) = str2num(yval);
    aline = fgetl(fid);
end

plot(xvec,yvec);
title([num2str(i), ' points']);

closeid = fclose(fid);
if closeid ~= -1
    fprintf('File closed successfully.\n');
else
    fprintf('File did not close successfully.\n');
end
```



```
>> WS9_3
File closed successfully.
>>
```

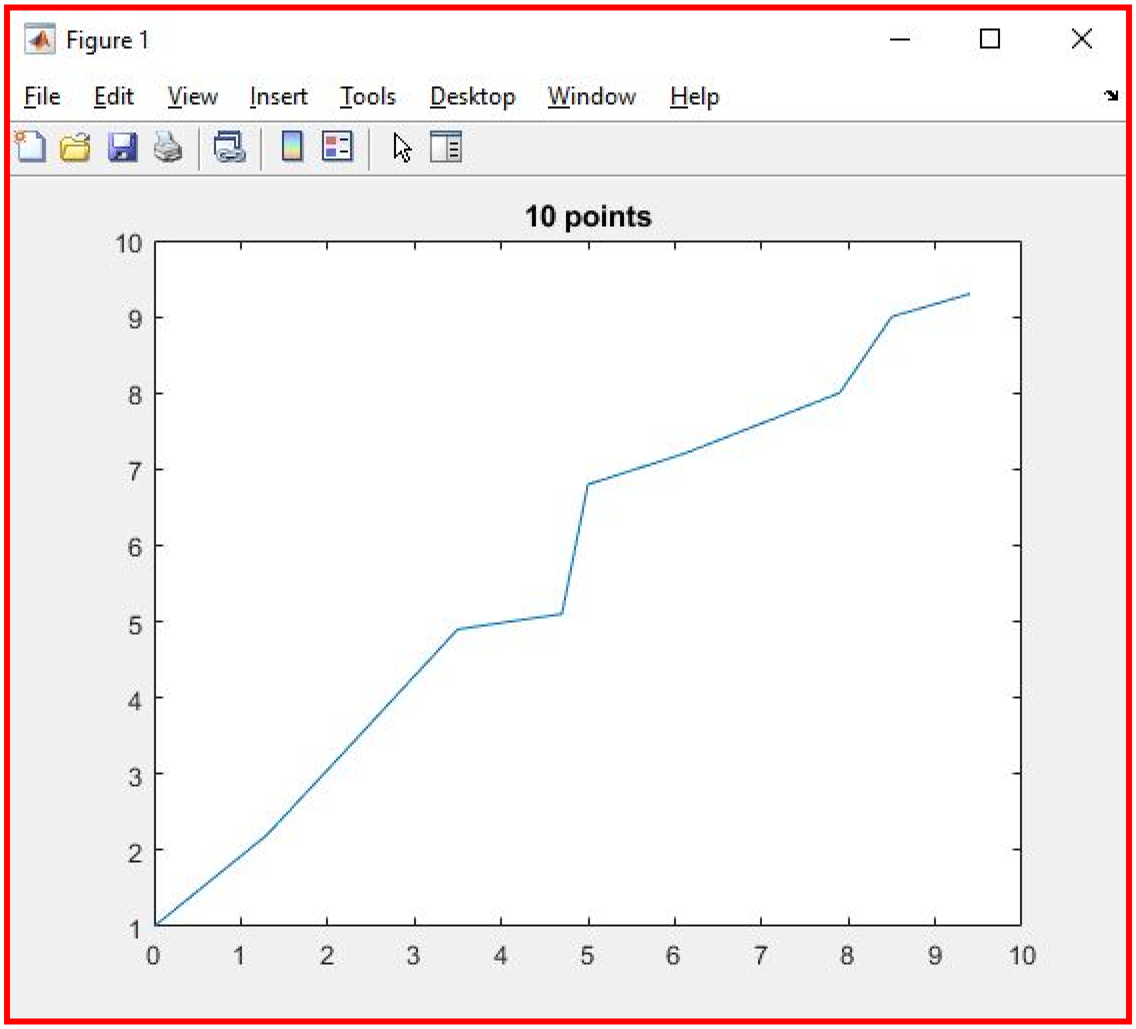
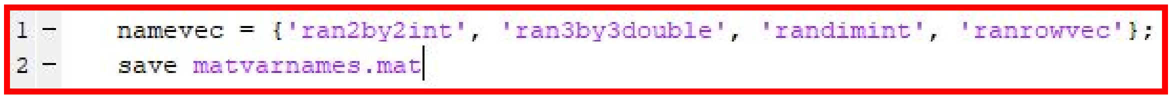
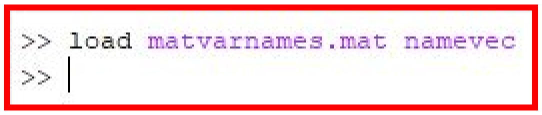


Figure 1
File Edit View Insert Tools Desktop Window Help
10 points
0 1 2 3 4 5 6 7 8 9 10
1 2 3 4 5 6 7 8 9 10

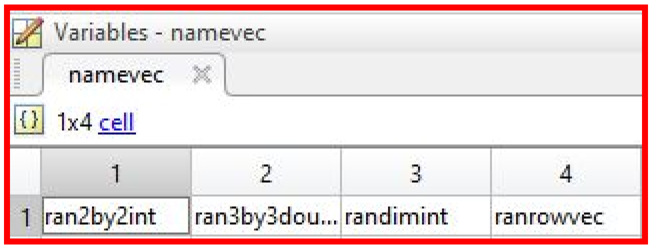
4) Create a set of random matrix variables with descriptive names (e.g. *ran2by2int*, *ran3by3double*, etc.) for use when testing matrix functions. Store all of these in a MAT-file.



```
namevec = {'ran2by2int', 'ran3by3double', 'randimint', 'ranrowvec'};
save matvarnames.mat
```



```
>> load matvarnames.mat namevec
>>
```



Variables - namevec

namevec

1x4 cell

	1	2	3	4
1	ran2by2int	ran3by3dou...	randimint	ranrowvec