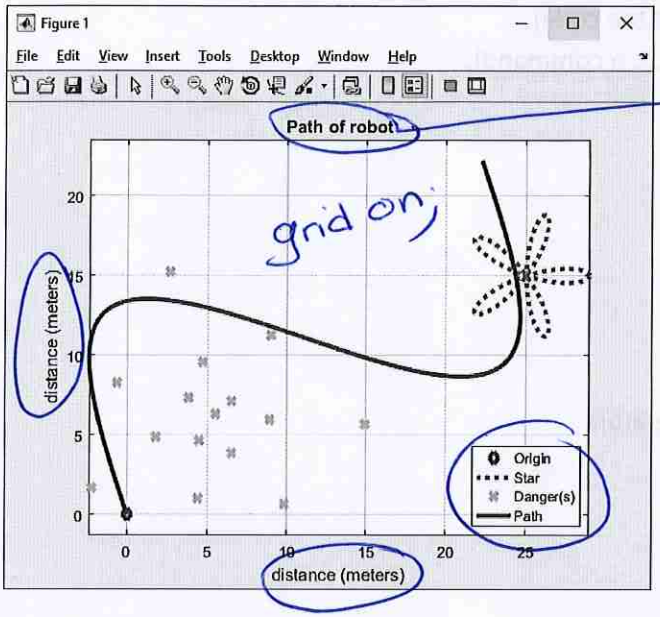
Name: ANSWER section: ____

Practice Exam1 – Variables, vectors, plotting

Guide the robot to the star! (About 35lines total, NO 7c applicable)
READ full directions please. Don't jump into code unless you know the end point!
Do as much as you can: do what you know first! COMMENT out anything that crashes but leave it! The basic rubric (and therefore points) can be done without actual code – do algorithm first – 10pts!

Also, to be clear: YOU MISS THE SUBMISSION LINK (55minutes after the hour), YOU GET A ZERO. – NO EXCEPTIONS.



You are to guide a robot to/through a final location (the star). It must avoid the danger "spots". Filename up to you.

Your graph should show all the data shown on the figure. Make sure to use different line styles for each plot. Different colors are optional. Nothing shows in the command window today.

1) The origin is at (0,0).

2) The star (y vs. x) is the following parametric equation:

$$x(t) = 2\cos(t) + 2\cos\left(\frac{2t}{3}\right) + 25$$
$$y(t) = 2\sin(t) - 2\sin\left(\frac{2t}{3}\right) + 15$$

where the parameter t goes from 0 to 6π.

cos() sin()
linspace(0, 6π)
OR 0:6*180
↳ cosd() sind()

3) The "dangers spots" (y vs. x) are also defined by another parametric equation:

$$x(\theta) = \theta * \cos(16\theta) + \frac{\cos(6\theta)}{2} + \frac{\sin(6\theta)}{3} + 5$$
$$y(\theta) = \theta * \sin(16\theta) + \frac{\sin(6\theta)}{2} + \frac{\cos(6\theta)}{3} + 6$$

where the parameter θ (theta) goes from 0 to 3π. **The number of data points in this vector of θ actually is the number of dangers spots! – this number of data points is a given that should be able to change easily.** When plotting this one, only use markers (no line).

linspace(0, 3π, N)

4) The path of the robot (y vs. x) is one final parametric equation:

$$x(\alpha) = \cos(\beta) * \alpha - \sin(\beta) * amplitude * \sin(T * \alpha)$$
$$y(\alpha) = \sin(\beta) * \alpha + \cos(\beta) * amplitude * \sin(T * \alpha)$$

Where the parameter α (alpha) goes from 0 to 10π. The other 3 givens (β *in degrees*, T, and *amplitude*) are scalars. (Besides angles, do not worry about units anywhere else).

Modify those givens (by trial and error) to avoid the danger spots the best you can and reach/go through the star.

Use test1 below as your starting point. When that works, reuse your code to fill in the blanks in the table!

	Number of danger spots	β (in degree)	T	Amplitude	Are dangers spots avoided? (Y/N)
Test1	15	45	0.20	10	Yes (figure shown)
Test2	47	(same as above)			NO
Test3	40	25	0.11	25	Yes

↑ I'm sure there are others.... LOL answers

*Do not zoom in anything. If the robot line touches a marker, it is not avoided.

Basic rubric:

Like pilots, this is your checklist. Remember – no credit for using what has not been taught. Show me the basic.

- ☐ 10pts The code you submit MUST RUN. NO CRASH. What does not run must be commented out (not deleted).

- ☐ 3pts valid filename
- ☐ 3pts your name, section, a description,
- ☐ 3pts appropriate clean up commands,
- ☐ 5pts an algorithm as comments (comments throughout the code),
- ☐ 5pts proper and valid variable names (test with the `which` command),
- ☐ 5pts semi-colon **every line,**
- ☐ 5pts proper and consistent spacing,
- ☐ 5pts no indent at this time, and
- ☐ 5pts correct figure, no output in command window

- ☐ 15pts data points x/y for the star
- ☐ 5pts given number of dangers spots can change easily
- ☐ 20pts data points x/y for the dangers
- ☐ 5pts proper/minimum use of element per element operators
- ☐ 5pts givens for robot can change easily
- ☐ 15pts data points x/y for the robot
- ☐ 5pts proper/minimum use of parentheses
- ☐ 10pts combined plots
- ☐ 10pts line types and markers
- ☐ 19pts all formatting on the figure
- ☐ 4pts table filled in
- ☐ 8pts leeway – anything you do that is wrong and was not expected.

(170pts)

```
%robot path to star
%caroline Liron, section XXX, spring 18

clc;
clear;
close all;
%format compact; %maybe

%star to reach
ts = linspace(0 , 6*pi);  %or 0:6*180, but then use cosd() sind()
xStar = 2*cos(ts) + 2*cos(2/3*ts) + 25;
yStar = 2*sin(ts) - 2*sin(2/3*ts) + 15;

%dangers zone
nPoints = 40;
thetas = linspace(0, 3*pi, nPoints);%colon not adequate here
xAvoid = thetas.*cos(16*thetas) + cos(6*thetas)/2 + sin(6*thetas)/3 + 5;
yAvoid = thetas.*sin(16*thetas) + sin(6*thetas)/2 + cos(6*thetas)/3 + 6;

%path of robot  (caution: alphas in randians, beta in degrees...)
alphas = linspace(0,10*pi);%radians, again 0:10*180, with cosd() sind()
betaControl = 25;
Tperiod = 0.11;
amplitude = 25;
xPath = cosd(betaControl)*alphas - sind(betaControl)*amplitude*sin(Tperiod*alphas);
yPath = sind(betaControl)*alphas + cosd(betaControl)*amplitude*sin(Tperiod*alphas);

%plot and format
plot(0,0,'d', xStar,yStar, ':', xAvoid,yAvoid,'x', xPath,yPath, 'linewidth',3);
axis equal;
legend('Origin','Star','Danger(s)','Path','location','best');
title('Path of robot');
xlabel('distance (meters)');
ylabel('distance (meters)');
grid on;

%the hold on; method is totally acceptable. I have the preference above as
%it is less LOC and I don't have to retype linewidth a billion time. but
%you do what you want. as long as I only see ONE hold on; command, you're
%ok.
```