BEE 4750 Homework 1: Introduction to Using Julia

Name:

ID:

> **Due Date**
>
> Thursday, 9/11/25, 9:00pm

Overview

Instructions

- Problem 1 consist of a series of code snippets for you to interpret and debug. You will be asked to identify relevant error(s) and fix the code.
- Problem 2 asks you to write code to implement several simple functions using more general programming and some which are unique to Julia (which might mean that you need to look at the documentation or use Julia's help function, which is accessed with `?`, e.g. `?sum` for help with the `sum()` function) or to analyze Julia syntax.
- Problem 3 asks you to convert a verbal description of a wastewater treatment system into a Julia function, and then to use that function to explore the impact of different wastewater allocation strategies.

Load Environment

The following code loads the environment and makes sure all needed packages are installed. This should be at the start of most Julia scripts.

```
import Pkg
Pkg.activate(@__DIR__)
Pkg.instantiate()
```

Standard Julia practice is to load all needed packages at the top of a file. If you need to load any additional packages in any assignments beyond those which are loaded by default, feel free to add a `using` statement, though you may need to install the package.

```
using Random
using Plots
using GraphRecipes
using LaTeXStrings
using Distributions
```

```
# this sets a random seed, which ensures reproducibility of random number generation. You sh
Random.seed!(1)
```

```
TaskLocalRNG()
```

Problems (Total: 30 Points)

Problem 1 (12 points)

The following subproblems all involve code snippets that require debugging. You are encouraged to use online resources (*e.g.* Julia documentation and forums, Stack Overflow, Reddit, etc) to help find diagnose error messages and find solutions, but make sure that you clearly document which resources you used and how you used them.

Problem 1.1

You've been tasked with writing code to identify the minimum value in an array. You cannot use a predefined function. Your colleague suggested the function below, but it does not return the minimum value.

```
function minimum(array)
    # initialize the minimum value counter
    min_value = 0
    # update minimum values
    for i in 1:length(array)
        if array[i] < min_value
            min_value = array[i]
        end
    end
    # return found minimum
    return min_value
```

```
end

array_values = [89, 90, 95, 100, 100, 78, 99, 98, 100, 95]
@show minimum(array_values);
```

```
minimum(array_values) = 0
```

What is the logical flaw in the code? Fix it and show that your fixed code solves the problem.

Problem 1.2

Your team is trying to compute the average grade for your class, but the following code produces an `UndefVarError`.

```
# enter student grade vector
student_grades = [89, 90, 95, 100, 100, 78, 99, 98, 100, 95]
# compute class average
function class_average(grades)
  average_grade = mean(student_grades)
  return average_grade
end

@show average_grade;
```

What does this `UndefVarError` mean? Why does this code produce one? Fix the error and solve the problem.

Problem 1.3

Your team wants to know the expected payout of an old Italian dice game called *passadieci* (which was analyzed by Galileo as one of the first examples of a rigorous study of probability). The goal of passadieci is to get at least an 11 from rolling three fair, six-sided dice. Your strategy is to compute the average wins from 1,000 trials, but the code you've written below produces a `MethodError`.

```
# function to simulate a passadieci roll: 3 6-sided dice
function passadieci()
    # this rand() call samples 3 values from the vector [1, 6]
    roll = rand(1:6, 3)
    return roll
```

```
end
# set number of trials and initialize outcome vector
n_trials = 1_000
outcomes = zero(n_trials)
# simulate number of passadieci rolls and count wins
for i = 1:n_trials
    outcomes[i] = (sum(passadieci()) > 11)
end
win_prob = sum(outcomes) / n_trials # compute average number of wins
@show win_prob;
```

What does this `MethodError` mean? Why does this code produce one? Fix the error and solve the problem.

Problem 1.4

You're interested in writing some code to remove the mean of a vector from all of its components. You've written the following code and tried to test it on a random vector, but your code returns a `MethodError`.

```
# function to remove mean from a vector
function remove_mean(vect)
    # fucntion to compute the mean
    function compute_mean(vect)
        element_sum = 0 # initialize sum
        # compute mean and return
        for v in vect
            element_sum += v
        end
        return element_sum / length(vect)
    end

    m = compute_mean(vect) # compute mean
    # return demeaned vector
    return vect - m
end

random_vect = rand(1_000)
@show remove_mean(random_vect)
```

What does this `MethodError` mean? Why does this code produce one? Fix the error and solve the problem.

Problem 2 (18 points)

Cheap Plastic Products, Inc. is operating a plant that produces 100m^3/day of wastewater that is discharged into Pristine Brook. The wastewater contains 1kg/m^3 of YUK, a toxic substance. The US Environmental Protection Agency has imposed an effluent standard on the plant prohibiting discharge of more than 20kg/day of YUK into Pristine Brook.

Cheap Plastic Products has analyzed two methods for reducing its discharges of YUK. Method 1 is land disposal, which costs $X_1^2/20$ dollars per day, where X_1 is the amount of wastewater disposed of on the land (m^3/day). With this method, 20% of the YUK applied to the land will eventually drain into the stream (*i.e.*, 80% of the YUK is removed by the soil).

Method 2 is a chemical treatment procedure which costs \$1.50 per m^3 of wastewater treated. The chemical treatment has an efficiency of $e = 1 - 0.005X_2$, where X_2 is the quantity of wastewater (m^3/day) treated. For example, if $X_2 = 50$m^3/day, then $e = 1 - 0.005(50) = 0.75$, so that 75% of the YUK is removed.

Cheap Plastic Products is wondering how to allocate their wastewater between these three disposal and treatment methods (land disposal, chemical treatment, and direct disposal) to meet the effluent standard while keeping costs manageable.

The flow of wastewater through this treatment system is shown in Figure 1. Modify the edge labels (by editing the `edge_labels` dictionary in the code producing Figure 1) to show how the wastewater allocations result in the final YUK discharge into Pristine Brook. For the `edge_label` dictionary, the tuple (i, j) corresponds to the arrow going from node i to node j. The syntax for any entry is `(i, j) => "label text"`, and the label text can include mathematical notation if the string is prefaced with an `L`, as in `L"x_1"` will produce x_1.

```
A = [0 1 1 1;
    0 0 0 1;
    0 0 0 1;
    0 0 0 0]

names = ["Plant", "Land Treatment", "Chem Treatment", "Pristine Brook"]
# modify this dictionary to add labels
edge_labels = Dict((1, 2) => "", (1,3) => "", (1, 4) => "",(2, 4) => "",(3, 4) => "")
shapes=[:hexagon, :rect, :rect, :hexagon]
xpos = [0, -1.5, -0.25, 1]
ypos = [1, 0, 0, -1]

p = graphplot(A, names=names,edgelabel=edge_labels, markersize=0.15, markershapes=shapes, ma
display(p)
```

Problem 2.1

Formulate a mathematical model for the treatment cost and the amount of YUK that will be discharged into Pristine Brook based on the wastewater allocations. This is best done with some equations and supporting text explaining the derivation. Make sure you include, as additional equations in the model, any needed constraints on relevant values. You can find some basics on writing mathematical equations using the LaTeX typesetting syntax here, and a cheatsheet with LaTeX commands can be found on the course website's Resources page.

Problem 2.2

Implement your systems model as a Julia function which computes the resulting YUK discharge and cost for a particular treatment plan. You can return multiple values from a function with a tuple, as in:

```
function multiple_return_values(x, y)
    return (x+y, x*y)
end

a, b = multiple_return_values(2, 5)
@show a;
@show b;
```

```
a = 7
b = 10
```

To evalute the function over vectors of inputs, you can *broadcast* the function by adding a decimal . before the function arguments and accessing the resulting values by writing a *comprehension* to loop over the individual outputs in the vector:

```
x = [1, 2, 3, 4, 5]
y = [6, 7, 8, 9, 10]

output = multiple_return_values.(x, y)
a = [out[1] for out in output]
b = [out[2] for out in output]
@show a;
@show b;
```

```
a = [7, 9, 11, 13, 15]
b = [6, 14, 24, 36, 50]
```

Make sure you comment your code appropriately to make it clear what is going on and why.

Problem 2.3

Use your function to experiment with 1,000 different combinations of wastewater discharge and treatment. You can do this with either a grid search or by sampling from a Dirichlet distribution (a Dirichlet$(1, n)$ distribution will generate uniformly-weighted n-dimensional vectors whose components add up to 1; see the `Distributions.jl` documentation for how to sample from probability distributions in Julia). Plot the results of these experiments. Do any satisfy the YUK effluent standard (plot this as well as a dashed red line). What was the cost of solutions satisfying the standard? What can you say about the tradeoff between treatment cost and YUK discharge? You don't have to find an "optimal" solution to this problem, but what do you think would be needed to find a better solution?

Problem 2.4

Find the strategies which minimize cost and YUK discharge (these will be different strategies) analytically and find the values of the objective metrics. Plot these values in the plot that you created for Problem 2.3. How do their values compare to the spread of values that you found in that problem? Would you select either of them (explain why or why not)?

References

List any external references consulted, including classmates.