BEE 4750 Homework 1: Introduction to Using Julia

Name: Anthony Nicolaides

ID: ajn68

> **Due Date**
>
> Friday, 9/8/23, 9:00pm

Overview

Instructions

- Problems 1-3 consist of a series of code snippets for you to interpret and debug. For Problems 1 and 2, you will be asked to identify relevant error(s) and fix the code. For Problem 3, the code works as intended; your goal is to identify the code's purpose by following its logic.
- Problem 4 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.

In [ ]:
```
import Pkg
Pkg.activate(@__DIR__)
Pkg.instantiate()
```

```
  Activating project at `~/Documents/BEE4750/hw/hw01-anthonynic28`
```

In [ ]:
```
using Plots
using GraphRecipes
using LaTeXStrings
```

Problems (Total: 40 Points)

Problem 1 (8 points)

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.

In [ ]:
```
function minimum(array)
    min_value = 0
    for i in 1:length(array)
        if array[i] < min_value
            min_value = array[i]
        end
    end
    return min_value
end

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

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

Problem 1.1 (3 points)

Describe the logic error.

The min_value variable is smaller than the minimum value in the array. Therefore, no value in the array gets declared to be the min_value, so min_value stays 0.

This can be resolved by either initalizing min_value to be the first number in the array (assumes it is a nonempty array), or to Inf (infinity).

Problem 1.2 (3 points)

Write a fixed version of the function.

In [ ]:
```
function minimum_fixed(array)
    min_value = Inf
    for i in 1:length(array)
        if array[i] < min_value
            min_value = array[i]
        end
    end
    return min_value
end
```

```
minimum_fixed (generic function with 1 method)
```

Problem 1.3 (2 points)

Use your fixed function to find the minimum value of `array_values`.

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

```
minimum_fixed(array_values) = 78
```

Problem 2 (8 points)

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

In [ ]:
```
student_grades = [89, 90, 95, 100, 100, 78, 99, 98, 100, 95]
function class_average(grades)
  average_grade = mean(student_grades)
  return average_grade
end

@show average_grade;
```

```
average_grade = 94.4
```

Problem 2.1 (3 points)

Describe the logic and/or syntax error.

First, the function is never called so no output from the function exists to assign it to average_grade, hence the error.

Second, student_grades is called from inside the function, making the argument 'grades' irrelevant, regardless if student_grades is a global variable.

Third, with the current packages imported, the mean() function is not defined, so the mean would need to be found manually.

Problem 2.2 (3 points)

Write a fixed version of the code.

In [ ]:
```
function class_average(grades)
  average_grade = sum(grades)/length(grades)
  return average_grade
end
```

```
class_average (generic function with 1 method)
```

Problem 2.3 (2 points)

Use your fixed code to compute the average grade for the class.

In [ ]:
```
student_grades = [89, 90, 95, 100, 100, 78, 99, 98, 100, 95]
average_grade = class_average(student_grades)
@show average_grade;
```

```
average_grade = 94.4
```

Problem 3 (8 points)

You've been handed some code to analyze. The original coder was not very considerate of other potential users: the function is called `mystery_function` and there are no comments explaining the purpose of the code. It appears to take in an array and return some numbers, and you've been assured that the code works as intended.

In [ ]:
```
function mystery_function(values)
    y = []
    for v in values
        if !(v in y)
            append!(y, v)
        end
    end
    return y
end

list_of_values = [1, 2, 3, 4, 3, 4, 2, 1]
@show mystery_function(list_of_values);
```

```
mystery_function(list_of_values) = Any[1, 2, 3, 4]
```

Problem 3.1 (4 points)

Explain the purpose of `mystery_function`.

mystery_function returns an array of values that contain the values of the inputed array, with the duplicates removed.

Problem 3.2 (4 points)

Add comments to the code, explaining why and how it works. Refer to "Best Practices for Writing Code Comments", and remember that bad comments can be just as bad as no comments at all. You do not need to add comments to every line (in fact, this is very bad practice), but you should note the *purpose* of every "section" of code, and add comments explaining any code sequences that you don't immediately understand.

In [ ]:
```
function mystery_function(values)
    y = []
    for v in values
        # puts values from array 'values' into array 'y'
        #  without repeating any values
        if !(v in y)
            append!(y, v)
        end
    end
    return y
end

list_of_values = [1, 2, 3, 4, 3, 4, 2, 1]
@show mystery_function(list_of_values);
```

```
mystery_function(list_of_values) = Any[1, 2, 3, 4]
```

Problem 4 (16 points)

Cheap Plastic Products, Inc. is operating a plant that produces $100\mathrm{m}^3/\mathrm{day}$ of wastewater that is discharged into Pristine Brook. The wastewater contains $1\mathrm{kg}/\mathrm{m}^3$ of YUK, a toxic substance. The US Environmental Protection Agency has imposed an effluent standard on the plant prohibiting discharge of more than $20\mathrm{kg}/\mathrm{day}$ 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 ($\mathrm{m}^3/\mathrm{day}$). 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 ($\mathrm{m}^3/\mathrm{day}$) treated. For example, if $X_2 = 50\mathrm{m}^3/\mathrm{day}$, 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, and chemical treatment, and land disposal) to meet the effluent standard while keeping costs manageable.

Problem 4.1 (3 points)

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.

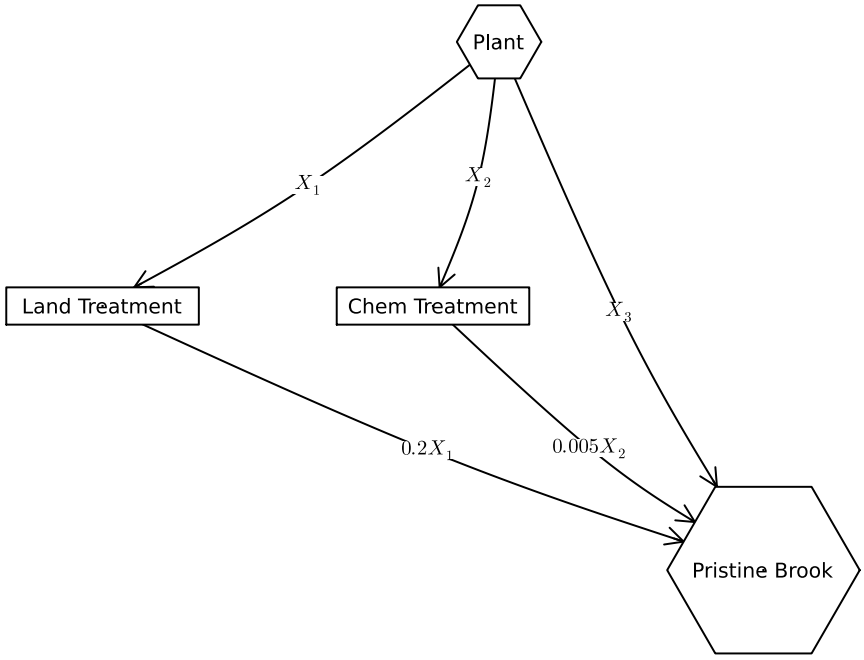
In [ ]:
```
using GraphRecipes, Plots

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) => L"X_1", #Plant ▶ Land Treatment
                   (1, 3) => L"X_2", #Plant ▶ Chem Treatment
                   (1, 4) => L"X_3", #Plant ▶ Pristine Brook
                   (2, 4) => L"0.2X_1", #Land Treatment ▶ Pristine Brook
                   (3, 4) => L"0.005X_2") #Chem Treatment ▶ Pristine Brook
```

```
shapes=[:hexagon, :rect, :rect, :hexagon]
xpos = [0, -1.5, -0.25, 1]
ypos = [1, 0, 0, -1]

graphplot(A, names=names,edgelabel=edge_labels, markersize=0.15,
    markershapes=shapes, markercolor=:white, x=xpos, y=ypos)
```



Problem 4.2 (4 points)

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.

Relevant values:

$100m^3/day$ of wastewater

$20kg/day$ of YUK can be discharged to the Pristine Brook, according to the EPA regulation

$1kg/m^3$ of YUK in the wasterwater (used for conversion between wastewater and YUK discharge)

$\$0.05X_1^2/day$ for Land treatment cost

$\$1.50/m^3$ for Chem treatment cost

X_1, X_2, X_3 all have units of wastewater, m^3/day

Constraints:

The plant releases $100m^3/day$ of wastewater which is shared between Land Treatment, Chem Treatment, and direct disposal, therefore:

$$X_1 + X_2 + X_3 = 100m^3/day$$

Legal regulation only allows at most 20 kg/day of YUK discharged into Pristine Brook, therefore:

$$TotalYUK <= 20kg/day$$

Equations:

Land Treatment, X_1, has 20 percent of YUK in wastewater be sent to the Pristine Brook. Chem Treatment, X_2, has 0.5 percent of YUK in wastewater be sent to the Pristine Brook. Direct disposal, X_3, has 100 percent of YUK in the wastewater be sent to the Pristine Brook. Therefore:

$TotalYUK = 0.2X_1 + 0.005X_2 + X_3\ [kg/day]$ (X_1, X_2, X_3 were converted to kg/day from m^3/day)

Land Treatment, X_1, cost $0.05X_1^2$ dollars per day. Chem Treatment, X_2, cost $1.50X_2$ dollars per day. Direct disposal, X_3, cost 0 dollars per day. Therefore:

$$Cost = 0.05X_1^2 + 1.50X_2\ [dollars/day]$$

Problem 4.3 (4 points)

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

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
```

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

In [ ]:
```
function wastewaterTreatment(x1, x2, x3)
    if ~((x1+x2+x3) == 100) # check to see if constaint is obeyed
        throw(ArgumentError("x1, x2, x3 must add up to 100"))
    end
    # calculate cost and total discharge of YUK
    totalYUK = (0.2*x1) + (0.005*x2) + x3
    cost = (0.05*(x1^2)) + (1.50 * x2)
    return (totalYUK, cost)
end
```

```
wastewaterTreatment (generic function with 1 method)
```

Problem 4.4 (5 points)

Use your function to experiment with some different combinations of wastewater discharge and treatment. Can you find one that satisfies the YUK effluent standard? What was the cost? 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?

In [ ]:
```
discharge, cost = wastewaterTreatment(45, 45, 10)
@show discharge;
@show cost;
```

```
discharge = 19.225
cost = 168.75
```

I found one combination that satifies the YUK effluent standard ($X_1 = 45, X_2 = 45, X_3 = 10$), and the cost was $168.75 To find a better solution, optimization would need to be implented. Gradient descent would reveal an optimal combination for the cheapest cost while maintaining the constraints.

References

List any external references consulted, including classmates.