BEE 4750 Homework 1: Introduction to Using Julia

Name: Fabien

ID: fmd48

> **Due Date**
>
> Thursday, 9/5/24, 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 gives you some code that works as intended; your goal is to identify the code's purpose by following its logic.
- Problem 3 asks you to write code to generate a random vector and subtract off its mean, using a Julia syntax called broadcasting.
- 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.
- Problem 5 (5750 only) asks you to use more advanced Julia techniques.

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 [1]:
```
import Pkg
Pkg.activate(@__DIR__)
Pkg.instantiate()
```

```
  Activating project at `c:\Users\FM's Laptop\Downloads\College\BEE 4750\HW\HW1\hw01`
   Installed GR_jll ──────────────────── v0.73.5+0
   Installed EarCut_jll ──────────────── v2.2.4+0
   Installed NetworkLayout ───────────── v0.4.6
   Installed ConcurrentUtilities ─────── v2.4.1
   Installed OffsetArrays ────────────── v1.14.0
   Installed ArnoldiMethod ───────────── v0.4.0
   Installed StaticArrays ────────────── v1.9.4
   Installed StaticArraysCore ────────── v1.4.2
   Installed GLFW_jll ────────────────── v3.3.9+0
   Installed Ratios ──────────────────── v0.4.5
   Installed IJulia ──────────────────── v1.24.2
   Installed TranscodingStreams ──────── v0.10.9
   Installed GraphRecipes ────────────── v0.5.13
   Installed CodecZlib ───────────────── v0.7.4
   Installed Xorg_libxcb_jll ─────────── v1.15.0+0
   Installed Conda ───────────────────── v1.10.0
   Installed TableTraits ─────────────── v1.0.1
   Installed GeometryTypes ───────────── v0.8.5
```

```
   Installed Plots ──────────────────── v1.40.4
   Installed Inflate ────────────────── v0.1.5
   Installed Graphs ─────────────────── v1.11.0
   Installed ZMQ ────────────────────── v1.2.5
   Installed Extents ────────────────── v0.1.2
   Installed ConstructionBase ───────── v1.5.5
   Installed DataValueInterfaces ────── v1.0.0
   Installed AbstractTrees ──────────── v0.4.5
   Installed GR ─────────────────────── v0.73.5
   Installed XSLT_jll ───────────────── v1.1.34+0
   Installed ChainRulesCore ─────────── v1.23.0
   Installed LogExpFunctions ────────── v0.3.27
   Installed SimpleTraits ───────────── v0.9.4
   Installed Adapt ──────────────────── v4.0.4
   Installed XML2_jll ───────────────── v2.12.7+0
   Installed IterTools ──────────────── v1.10.0
   Installed Interpolations ─────────── v0.15.1
   Installed Latexify ───────────────── v0.16.3
   Installed OpenSSL_jll ────────────── v3.0.13+1
   Installed BitFlags ───────────────── v0.1.8
   Installed Tables ─────────────────── v1.11.1
   Installed GeometryBasics ─────────── v0.4.11
   Installed IteratorInterfaceExtensions ─ v1.0.0
   Installed Qt6Base_jll ────────────── v6.5.3+1
   Installed StructArrays ───────────── v0.6.18
   Installed AxisAlgorithms ─────────── v1.1.0
   Installed WoodburyMatrices ───────── v1.0.0
   Installed UnitfulLatexify ────────── v1.6.3
   Installed GeoInterface ───────────── v1.3.4
    Building Conda ─→ `C:\Users\FM's Laptop\.julia\scratchspaces\44cfe95a-1eb2-52ea-b672
-e2afdf69b78f\51cab8e982c5b598eea9c8ceaced4b58d9dd37c9\build.log`
    Building IJulia → `C:\Users\FM's Laptop\.julia\scratchspaces\44cfe95a-1eb2-52ea-b672
-e2afdf69b78f\47ac8cc196b81001a711f4b2c12c97372338f00c\build.log`
Precompiling project...
  ✓ IteratorInterfaceExtensions
  ✓ DataValueInterfaces
  ✓ IterTools
  ✓ ConcurrentUtilities
  ✓ AbstractTrees
  ✓ Extents
  ✓ OffsetArrays
  ✓ Ratios
  ✓ TranscodingStreams
  ✓ Inflate
  ✓ ChainRulesCore
  ✓ Adapt
  ✓ ConstructionBase
  ✓ BitFlags
  ✓ StaticArraysCore
  ✓ WoodburyMatrices
  ✓ LogExpFunctions
  ✓ Latexify
  ✓ OpenSSL_jll
  ✓ SimpleTraits
  ✓ EarCut_jll
  ✓ XML2_jll
  ✓ TableTraits
  ✓ Conda
  ✓ Ratios → RatiosFixedPointNumbersExt
  ✓ TranscodingStreams → TestExt
  ✓ GeoInterface
  ✓ ChainRulesCore → ChainRulesCoreSparseArraysExt
  ✓ OffsetArrays → OffsetArraysAdaptExt
  ✓ Unitful → ConstructionBaseUnitfulExt
  ✓ AxisAlgorithms
  ✓ ZMQ
```

```
  ✓ Gettext_jll
  ✓ XSLT_jll
  ✓ LogExpFunctions → LogExpFunctionsChainRulesCoreExt
  ✓ CodecZlib
  ✓ Tables
  ✓ OpenSSL
  ✓ Xorg_libxcb_jll
  ✓ Glib_jll
  ✓ UnitfulLatexify
  ✓ GLFW_jll
  ✓ IJulia
  ✓ Cairo_jll
  ✓ Qt6Base_jll
  ✓ StatsBase
  ✓ HarfBuzz_jll
  ✓ libass_jll
  ✓ HTTP
  ✓ StaticArrays
  ✓ FFMPEG_jll
  ✓ StaticArrays → StaticArraysChainRulesCoreExt
  ✓ Adapt → AdaptStaticArraysExt
  ✓ ConstructionBase → ConstructionBaseStaticArraysExt
  ✓ StaticArrays → StaticArraysStatisticsExt
  ✓ FFMPEG
  ✓ StructArrays
  ✓ GR_jll
  ✓ StructArrays → StructArraysAdaptExt
  ✓ StructArrays → StructArraysSparseArraysExt
  ✓ ArnoldiMethod
  ✓ StructArrays → StructArraysStaticArraysExt
  ✓ Interpolations
  ✓ GeometryTypes
  ✓ Interpolations → InterpolationsUnitfulExt
  ✓ GR
  ✓ Graphs
  ✓ GeometryBasics
  ✓ NetworkLayout
  ✓ NetworkLayout → NetworkLayoutGraphsExt
  ✓ GraphRecipes
  ✓ Plots
  ✓ Plots → UnitfulExt
  ✓ Plots → IJuliaExt
  ✓ Plots → GeometryBasicsExt
  75 dependencies successfully precompiled in 186 seconds. 126 already precompiled.
```

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.

In [4]:
```
using Random
using Plots
using GraphRecipes
using LaTeXStrings
```

In [3]:
```
# this sets a random seed, which ensures reproducibility of random number generation. Yo
Random.seed!(1)
```

```
TaskLocalRNG()
```

Problems (Total: 50/60 Points)

Problem 1 (15 points)

The following subproblems all involve code snippets that require debugging.

For each of them:

- identify and describe the logic and/or syntax error;
- write a fixed version of the function;
- use your fixed function to solve the problem.

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.

In [11]:
```
function minimum(array)
    min_value = array[1] #needs to be first value in array
    for i in 2:length(array) # start from the second element for the loop
        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) = 78
```

Small fix for this is to change where the min_value starts which is supposed to be the first value in the array and then the loop is suspposed to start from the second element in the loop.

Problem 1.2

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

In [16]:
```
#Out of order in the syntax and calling the wrong function. Should not be calling averag
function class_average(grades)
  average_grade = sum(grades)/length(grades)
  return average_grade
end
student_grades = [89, 90, 95, 100, 100, 78, 99, 98, 100, 95]
@show class_average(student_grades);
```

```
class_average(student_grades) = 94.4
```

For this fucntion class_average should be called not average_grade in the last line

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 an error.

In [25]:
```
function passadieci()
    # this rand() call samples 3 values from the vector [1, 6]
    roll = rand(1:6, 3)
    return roll
end
```

```
n_trials = 1000
outcomes = zeros(n_trials)# should be zeros instead of zero
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;
```

```
win_prob = 0.374
```

The main error was the zero needed to be altered to zeros to run properly in line 7.

Problem 2 (5 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 [26]:
```
function mystery_function(values) #returns a list removing duplicate values
    y = []
    for v in values
        if !(v in y) # Checks if v is not in already in y
            append!(y, v) # if it is not in y append it
        end
    end
    return y
end

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



#the unique function also does the same thing
list_of_values = [1, 2, 3, 4, 3, 4, 2, 1]
unique_values = unique(list_of_values)
println(unique_values) #Output: [1, 2, 3, 4]
```

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

For this function the code was meant to remove the duplicates form a list of values. There is a similar function in Julia called unique() that also does the same thing.

In this problem:

- Explain the purpose of `mystery_function`.
- 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.
- Is there a built-in Julia function that does the same thing as `mystery_function`? If so, what is it? Use it to do the same thing as the code above.

Problem 3 (10 points)

You're interested in writing some code to remove the mean of a vector.

In this problem:

- Write a function `compute_mean(v)` which sums all of the elements of a vector `v` using a `for` loop and computes the mean.
- Make a random vector `random_vect` of length 10 using Julia's `rand()` function. Use your `compute_mean()` function to calculate its mean and subtract it from `random_vect` **without a loop** (using a Julia technique called *broadcasting*; feel free to consult the Julia documentation and search as necessary). Check that the new vector has mean zero.

In [142…

```
function compute_mean(v)
  count=0
  n=length(v)
  for i in v
    count += i
  end

  mean=count/n
  return mean
  # mean = sum(v)/length(v);
  #   return mean
  # end
end


random_vect = rand(10);
mew = compute_mean(random_vect);
newvec = random_vect.-mew;
check = compute_mean(newvec);
println(check)

if isapprox(check, 0)
  println("The result is approximately zero.")
else
  print("The result is not zero.")
end
```

```
0.0
The result is approximately zero.
```

For this function the mean was computed first using a count to retain all the values in v and then summing up the values and dividing by the amount of numbers summed up. Next a random vector was created with size 10 and the mean function was run on it. The mean was subtracted from the new random vector with size 10 and a check was run to make sure the code result for the mea on the new vector was zero.

Problem 4 (20 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).

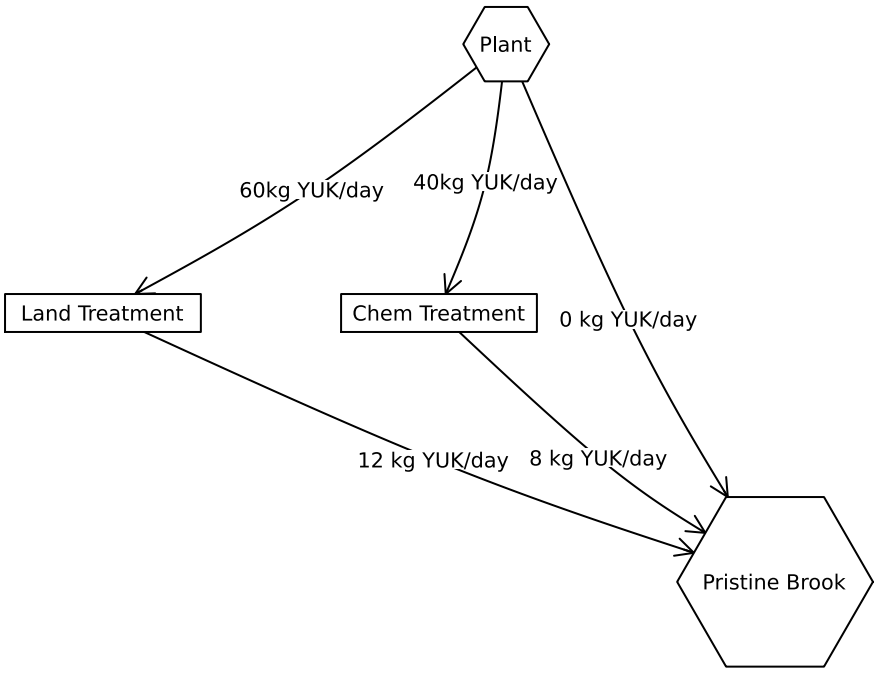
Method 2 is a chemical treatment procedure which costs \$1.50 per m^3 of wastewater treated. The chemical treatment has an efficiency of $e = 1 - 0.005X_2$, where X_2 is the quantity of wastewater ($\mathrm{m}^3/\mathrm{day}$) treated. For example, if $X_2 = 50\mathrm{m}^3/\mathrm{day}$, then $e = 1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

In [182…

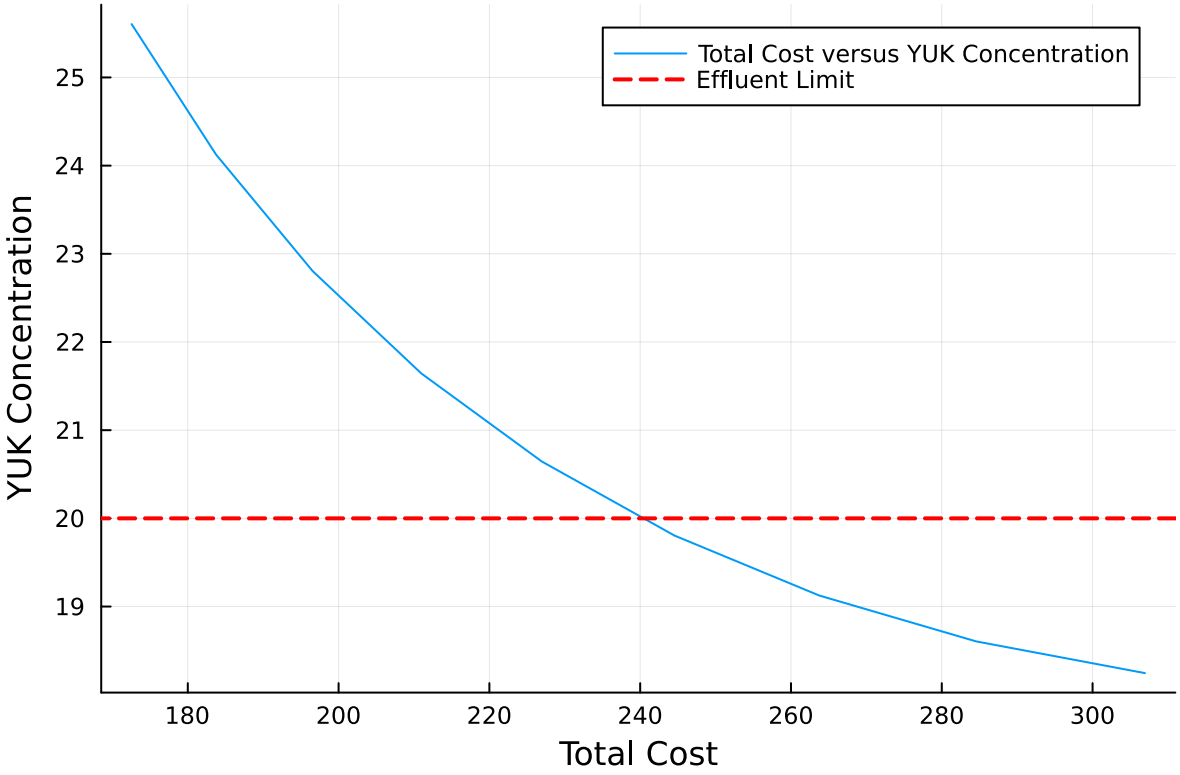
```
A = [0 1 1 1;
    0 0 0 1;
    0 0 0 1;
    0 0 0 0]

names = ["Plant", "Land Treatment", "Chem Treatment", "Pristine Brook"]
# modify this dictionary to add labels
edge_labels = Dict((1, 2) => "60kg YUK/day", (1,3) => "40kg YUK/day", (1, 4) => "0 kg YU
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
display(p)
```



The code below is created to function as a formula to find the land and chemical treatment output and the plant output

In [7]:
```
#land+chem+direct = 100kg Yuk /day
#function takes in vlaues from the land treatment option and chem treatment
#Finds the total cost of operation and YUK concentration total entering Pristing Brook

function Yuk_output(land,chem)

    #finding costs of operations for each method
    cost1 = ((land)^2)/20 #in kg/$/d
    cost2 = 1.5*chem
    cost3 = 0
    total_cost = cost1 + cost2 + cost3

    #find YUK concentration
    YUK1 =  land - (0.8 * land)
    YUK2 = (chem^2) *0.005
    YUK3 = 100 - (land + chem)


    YUK_Total = YUK1 + YUK2 +YUK3
    # println("Land output: ",YUK1)
    # println("Chem output: ", YUK2)
    # println("Plant output: ",YUK3)
    return total_cost,YUK_Total
end

#returns the total_cost and the YUK_Total
total_cost,YUK_Total = Yuk_output(60,40)
# println(total_cost)
# println(YUK_Total)

#test
land = [41,45,49,53,57,61,65,69,73]
chem = 100 .-land
#runs function on multiple inputs
YUK_conc = Yuk_output.(land,chem)

#returns output values of land and chem
TotalCost = [out[1] for out in YUK_conc]
YUKconc = [out[2] for out in YUK_conc]
@show TotalCost;
@show YUKconc;

#Create Scatter to show the range of values for the test
pl = plot(TotalCost,YUKconc,xlabel = "Total Cost" , ylabel = "YUK Concentration", label
hline!([20], label= "Effluent Limit", color=:red, linestyle=:dash, linewidth=2)
```

```
TotalCost = [172.55, 183.75, 196.55, 210.95, 226.95, 244.55, 263.75, 284.55, 306.95]
YUKconc = [25.604999999999997, 24.125, 22.805, 21.644999999999996, 20.645, 19.8049999999
99996, 19.125, 18.604999999999997, 18.244999999999994]
```



In this problem:

- 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.

- Implement your 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:

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

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

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:

```
{julia}
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;
```

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

- Use your function to experiment with some different combinations of wastewater discharge and treatment and plot the results of these experiments. Can you find one that satisfies the YUK effluent standard (plot this as well as a dashed red line)? What was the cost? What can you say about the tradeoff between treatment cost and YUK concentration? 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?

The tradeoff is that as the YUK Concentration gets smaller the cost increases. The optimal cost is around 60 kg/day of wastewater going into the land treatement and about 40 kg/day of wastewater in the chem treatment. To find a better solution there should be an optimized or set value for the three systems.

References

List any external references consulted, including classmates.