BEE 4750 Homework 1: Introduction to Using Julia

Name: Yingying LIU

ID: 47236769

> **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.

In [2]:
```
import Pkg
Pkg.activate(@__DIR__)
Pkg.instantiate()
```

```
  Activating project at `C:\Users\Lyy\lab1-132140636yyy\hw01`
```

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 [3]:
```
using Random
using Plots
using GraphRecipes
using LaTeXStrings
using Distributions
```

In [4]:
```
# this sets a random seed, which ensures reproducibility of random number generation. You should always
Random.seed!(1)
```

Out[4]:
```
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.

Answer 1.1

When I first tested this code, it returned 0 as the minimum value, even though 0 does not exist in the provided array. The actual minimum should be 78. I suspect the error occurs because **the initial min_value was too small and it was set to 0**, so no element in the array is smaller than this starting value.

To fix the problem, we need to initialize min_value **with a sufficiently large number**. I **changed it from 0 to 10,000,000**, and now the minimum() function correctly returns 78.

In [ ]:
```
function minimum(array)
    # initialize the minimum value counter
    min_value = 10000000
    # 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) = 78
```

Problem 1.2

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

Answer 1.2

The meaning of `UndefVarError` is it **cannot find the correspongding function after @show**. Since the function name class_average() is missing, this code cannot successfully be run. Also we should put the array student_grades within (), instead of average_grade. Average_grade is just a defined variable within function class_average.

So the correct code should be **@show class_average(student_grades);**.

In [8]:
```
# 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 class_average(student_grades);
```

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

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`.

Answer 1.3

The MethodError occurs because **zero(n_trials) was used incorrectly**. What we actually need is **an array of length 1000 to store the generated random values**. However, zero(n_trials) only returns the integer 0, not an array. Since an integer cannot hold 1000 values, the code fails when attempting to index into it.

In [16]:
```
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 = zeros(Int,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;
```

```
win_prob = 0.391
```

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`.

Answer 1.4

The MethodError occurs because **vect is a vector and m is a scalar**, so they cannot be subtracted directly. If we want to subtract m from each element of vect, **we need to use the elementwise operator .- instead of -**.

In [24]:
```
# 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)
```

remove_mean(random_vect) = [0.037127131801273694, -0.3579116316050869, 0.4558194989445373, -0.3606997512
1248155, 0.25279440864812697, -0.4874824481803701, 0.4875370394219454, 0.4406193193986143, -0.3894187894
165013, 0.3902347729979515, 0.24510466748580828, 0.2410973132164358, 0.015587271219666077, 0.45806045539
03488, 0.4611588650745191, 0.40105021657053064, 0.017313652795339918, -0.3398158559065353, -0.1114583064
6778099, -0.27640209024812434, 0.44810456250654507, 0.4284427123205996, -0.10266484446402135, 0.01914375
0753844224, 0.34793613943742574, -0.26765074532067823, 0.415937762657283, 0.10738675395636166, 0.4357968
4405197416, -0.493669538520353, 0.3176573709816213, -0.02769873805946932, -0.04215974468023176, 0.349540
80045577884, -0.025164733232810432, -0.334331540728945, 0.09976192037701304, -0.34089947997336045, 0.083
88849242975949, 0.07036294081554162, 0.23852257818280453, -0.3809340422850135, 0.1818960553897897, -0.03
401682705386322, -0.33907546701780444, 0.2779696930982487, -0.4036858207159214, 0.26134425883369805, 0.4
920388074992108, -0.4778157618709339, -0.19175106216052085, -0.1834991338561015, 0.2906417753504418, -0.
16364814960308693, 0.1769057639949979, 0.0638535334635546, -0.23889096031557266, -0.3463861619011418, -
0.374102410798537, -0.49104754372582327, 0.3037918882738532, 0.3238561237941191, 0.19085062300093047, -
0.22214074464104294, -0.14792814573619217, 0.10897288253943938, -0.07446648827468305, -0.491367052949178
8, -0.3364408306669414, 0.007666877770928449, -0.1127943949669109, 0.17682008575607033, 0.00771631907894
5544, -0.3480492654879447, 0.01124517474120934, -0.1868650543947339, -0.49132214473990643, 0.38958645675
312853, -0.35052802495145474, 0.3500273298185067, -0.4589778572385672, -0.013784754696009816, 0.08301316
964182315, 0.46147196703999716, -0.366910311989857, -0.01537756437654314, -0.11966858916842515, -0.38433
083830393155, 0.09877130288380132, 0.44526140241088974, -0.07736439215759783, 0.22632826249593285, -0.49
87979530754525, -0.40065589673589463, -0.03816701353650542, 0.23772435370133638, -0.027993389966215032,
-0.08389822387691248, 0.3284534879816309, -0.043494141717075885, -0.2711253107291246, -0.189894634122558
14, -0.3290502550903621, -0.26727990314750094, 0.36574962035833014, -0.3673076788715831, -0.218037457680
54839, -0.0014365641396002493, 0.09476984476216044, 0.012493060119654342, -0.13120736479173112, -0.36484
61345725753, 0.4360373426957027, -0.332403270362731, -0.07679710912300997, 0.05212392049876724, 0.389944
79114915204, -0.46762931302273536, -0.052513252870646676, 0.13016865480147588, 0.23795171285860683, 0.08
101052115712593, -0.26507679856435173, 0.06110152973981675, -0.04786337067531299, -0.4261150569559413,
0.39581941483105476, -0.40933692222934426, 0.22810139430801346, -0.49168575063179676, -0.022565565165170
72, -0.06872982343742429, -0.27924847616810766, 0.2423311839467177, 0.3620002686768774, -0.0487876732990
22005, -0.40376544200861963, 0.22825821604712637, 0.020402572270616925, -0.13107115103210287, -0.1929094
6499967698, 0.4231933802724377, -0.18770940426156502, 0.47878863570608476, -0.18779800954215353, 0.39737
793304960467, -0.1813436212373598, -0.18159383348225688, 0.12910228719702455, -0.029101877606904747, 0.0
5225683125341629, 0.23755765719376076, -0.37546340421616253, 0.1684809129885011, 0.05202159090243019, 0.
13913022098689065, -0.018704926858530535, -0.16846326030514724, 0.061536480317689546, -0.065229004444871
56, 0.0001671052446385879, -0.13963672062541987, -0.057593869016560495, -0.46603686483667783, 0.21550583
07565635, -0.08175300938999952, -0.2756265171813803, 0.06391974153839486, 0.09972532184699101, -0.014386
494403598182, -0.02065717949166912, -0.44291494305510604, -0.4728902421719323, -0.4451090408637264, 0.16
532740627446896, 0.15277359334667573, 0.02579969878830779, 0.400442148112198, -0.24815643887446348, 0.24
917723331870245, -0.2545159996285218, -0.04989872096126713, 0.09008606456634216, -0.4789589966879134, -
0.29554699831520426, -0.17680023355345043, -0.11412996986147028, 0.39559422879244477, 0.168912557692233
9, 0.3433386944977642, 0.1699273367319506, 0.07194509538391591, -0.2321819631304115, -0.1178692191728352
2, 0.45139291822467364, 0.32776651972997295, 0.15925867538788452, 0.08956048260138838, 0.434998423750771
36, 0.4349433872801698, -0.18725680770769015, -0.10677129765893001, 0.3251208548019161, 0.15479828189833
067, -0.06344368429022318, -0.3104419689928204, 0.1401032691336629, 0.45664918768568297, -0.237871578433
32307, 0.3397978111301382, -0.4027552004296997, -0.03577405071814943, -0.2156299976167909, -0.3164635259
5665694, 0.23828530527969793, -0.14780596455019712, -0.10742639490402728, -0.0722378127501172, -0.098238
11873551247, -0.09867968073649402, -0.08680117747415983, -0.46999313267582365, -0.33144281451492963, -0.
12041654626405685, -0.16560759909704892, -0.023219213944494665, -0.1588599881827506, 0.01006321389945363
8, -0.11921246684873088, 0.44590087433032244, -0.15288561041842008, -0.3873041915597152, 0.4795729752557
336, 0.23974329815104545, 0.3366663418949648, 0.1494687942272167, -0.417336188867504, 0.1765661959070444
8, -0.30672185269801355, 0.33460531427112117, 0.18722512340744768, -0.18804394703902116, 0.1797532535339
7864, 0.21201935785245962, 0.367614895071563, -0.20576528828194063, -0.2454013174031734, -0.484458619223
84126, 0.3122580017503207, 0.2997592751944167, -0.015635394612461218, 0.3849212424958629, 0.190521867464
723, 0.3889751429596783, 0.24277243267677984, 0.12234484203859464, 0.27519221931207405, -0.4416294122820
2623, 0.10609568188653229, -0.05073270860895884, 0.06944118589836579, 0.384449151220565, 0.0856992423174
0667, 0.4469235969581833, 0.1446907263827183, 0.4233200794022258, -0.24607684441366218, 0.15576583436856
295, -0.030278975352423187, 0.03149532368718033, -0.27990652120503556, 0.30317236897922817, 0.3330427900
323696, 0.25909757388723287, -0.28180665495587653, -0.28202028855618144, 0.1758498321741332, 0.013048271
617988827, 0.4461089310780473, 0.027266881872623494, 0.20321222006496575, -0.32742734638615956, -0.46668
00009909172, -0.39404773499582546, -0.007527891627368111, -0.26097360678222103, 0.27647120265602876, -0.
3110414597391088, 0.3240223279236343, 0.1511124034891297, 0.4832420989013064, -0.4769527649733526, -0.14
674003797694446, -0.4515554982027895, 0.43893584936468366, -0.07855806532045184, 0.38287741989602875, 0.
08929861842782494, -0.25995554115749697, -0.4889816922318918, 0.028017556674170496, 0.02707410773959839
6, -0.06058803315660022, 0.2718242072319973, 0.4025116558120386, 0.24699934909322552, -0.049848029786473
4, 0.12054505802383442, -0.34381598369360433, -0.036028102537441486, -0.062247223301760246, 0.3776831289
3099654, -0.3725509153932487, -0.15153991656264743, 0.371231827565622, -0.024959334946298206, -0.0829214
300799197, -0.08580899787882O9, -0.363304236369816, -0.16695172922751167, -0.4905975683606395, 0.1170377
0853372408, 0.16009777977824102, -0.3807497718014955, 0.16318354763036225, -0.39441222138982357, 0.16195
678050815887, 0.0627892267470681, -0.3388594771007587, -0.055934708372607056, -0.1365696680584383, -0.35
93012344808859, -0.4096760668682339, 0.1542267542637038, -0.23534896717702614, 0.18378055378248304, 0.00
1689520260092836, 0.4248931341834723, -0.27872052127687563, 0.22487196895816275, 0.22014083149850894, -
0.0834445479824234, -0.0487007748078917, 0.35486775589118247, 0.17108215750309275, 0.40300172094275033,

0.40174036886905873, 0.11740824902513303, 0.013202792985930523, -0.2904072598468178, -0.326382541137203, -0.3361552489442149, -0.49934235691916573, -0.3647191989483376, -0.2957588285360889, 0.057062421945034636, 0.3775059120468358, -0.0007664777034277481, -0.10647489555993839, 0.19772161226518825, 0.10746828474191461, 0.382122156074687, 0.0014402749734609976, 0.3203982617032193, 0.057803349247765756, 0.38602460186509124, -0.09151753049981715, -0.1392653951879652, 0.4422217494511007, -0.38553684160506885, 0.38003051079458483, -0.12714261677359284, 0.4129242826776319, -0.2275380166959151, -0.18474211158357534, 0.44069425576746524, -0.19899798063848795, 0.34018020937788007, 0.16131302989782448, 0.2738452298362941, 0.473827161798541, 0.19251813996690947, 0.11532320810501562, 0.16207220126905375, 0.16643296117935213, -0.5012169392209032, 0.45569615553823517, 0.24526365700393837, -0.33730657464171443, -0.191426402587478, -0.30395087481466354, 0.4648970620645134, -0.4639280392413626, -0.4911984731967667, -0.25095999647274403, 0.3384817090100882, -0.24374899345944778, 0.2536045611520128, 0.28269321348225707, 0.38428532831864337, -0.32103068934938417, -0.18938732576506268, 0.4310279486475811, -0.3285953903438046, 0.38011365329905367, -0.19674831220981792, 0.25388481652005757, 0.15190717406381737, 0.3386589238019292, -0.12141620025309274, 0.18043250584366666, -0.14488608750357512, -0.013229754388511372, 0.15765154210072718, 0.0685210963400398 9, 0.24561895175533244, 0.16373698168160666, 0.2754402258501115, -0.27575244732146265, -0.46476999572534394, -0.39667492164488893, -0.5057797693371403, 0.397838309901552, 0.4149983231624331, -0.06417574971895912, -0.05880087853775795, 0.2689823201634072, 0.2594067889170055, -0.3808266131589666, -0.20955367713569095, 0.08101457919398969, 0.22132528633851722, 0.4146156095401552, -0.061458423612807 29, 0.11381939415383124, 0.11849146517728326, 0.1840640490652382, 0.4723345672094138, 0.22984154097100906, 0.09487458884337663, 0.30979068558875966, 0.13375956029774516, 0.29291313289042487, 0.45115072285250235, 0.06970103606568456, 0.15812935340464562, 0.048967709223936784, 0.3798357115667468, 0.48068899793287156, 0.3022854136096388, 0.20697454928315662, -0.24467643307988274, 0.1424049139417396, -0.07169727050748054, -0.01018718861783019, -0.22613075539393512, 0.15677227236303937, 0.33738919724393357, 0.298286740221938, 0.48045636564686856, 0.09388390625824794, -0.1029355436797269, -0.42462605174748835, 0.076243571636181, 0.10679695414025026, 0.43105987170184057, 0.28862727132121335, 0.07807747610400095, 0.4509366066890934, -0.5043415242030906, -0.46995121926842176, -0.2187694262400669, -0.29597328044178817, -0.2673758075033207, -0.3703419267564265, -0.2245963228027198, -0.502966582298158, 0.41779793811903854, 0.47095596743868484, 0.3429751925445187, 0.09394689099374742, -0.07824795743629576, -0.3927198708060562, 0.19198393682786385, -0.26016845284385925, 0.02745291095742164, -0.036697884084100685, -0.07258813446589618, 0.24810379486993572, -0.12831456611410041, -0.16864727239198407, -0.26354711365426886, 0.30805373575477435, -0.3105944167002318, -0.41337415744908657, -0.40601307607086257, 0.06288835665732417, 0.10184016678724506, -0.2734286074014376, -0.14281609126574113, 0.4386665332644837, 0.14335442514971275, 0.3471798583142053, 0.19076001218959893, -0.4741146964036411, 0.39050836170593606, -0.08947839670614899, -0.2004347284503568, 0.24914996455328386, 0.021493246733649274, -0.3268975087707583, -0.45872539792042566, 0.44133338275074596, 0.025478330143544503, 0.2505869652083462, 0.4058503850060715, -0.36045684312641213, 0.07583577505325345, 0.08387059372276684, 0.3919082154357296, 0.3164865443706091, 0.04061213229191396, -0.3756162407095609, -0.19009057431206489, -0.4479000737545089, -0.3492112273470612, -0.49613531153981116, -0.3801121265294666, -0.43062072172439714, -0.34157799369400255, -0.4020007378024303, 0.010964070819634797, -0.07336580636223067, -0.3644558147988556, -0.019427539054437726, 0.1234336915292249, 0.04288684095417217, 0.27419609970849523, -0.07135109070141799, -0.2398919571133492, 0.40475955730461133, -0.3614441306817139, -0.19702439107613545, -0.31194900029172945, 0.33545292057163634, -0.26183254583905, 0.030353546246230212, -0.47827782188400747, -0.3025706552355333, 0.3722517181686106, 0.0572753022505853, -0.020052589115656883, 0.4885816450007838, 0.4211729396455898, 0.3735066037646092, -0.3261409053515827, 0.45976857787620307, -0.1257232869359478, 0.36282244501054417, 0.18556648863984326, -0.4142429796335102, -0.42907686274856527, -0.07085059936462457, -0.3396218281104827, -0.19965486285556988, 0.28469577923099476, 0.34201732755275827, -0.4878295087775193, -0.23648831180011987, -0.15709189690559944, -0.488915720105791, -0.03394907531490887, -0.35775919188690264, 0.03203232055929239, -0.2220212714050276, -0.4401351593131335, -0.4644571658667461, 0.28078414407427355, -0.24116583604561537, -0.47759118774654485, -0.06008684642127293, -0.21732431018640597, -0.04629541836512496, -0.2942244773766596, -0.21274362759643062, -0.355039919347544, 0.39339849320604636, -0.08921620437264277, -0.14516214979513076, -0.24503758370090434, -0.033987268507489454, 0.32486118479914017, 0.187634875374861, -0.007956474677116243, -0.18836969475751042, 0.2799065419059682, -0.2838785103771385, 0.28284123482337775, 0.47632440579073554, -0.37102193519573123, -0.07099955181839257, 0.41213860009174751, -0.1691357294846454, 0.4456905103232912, -0.16583145814559996, 0.14429434886357384, -0.18941080691661816, -0.22979042202466016, -0.19841389718576996, 0.11152889416240164, 0.11086401929219869, -0.292651744215114, 0.46373889386326006, -0.46308484590964105, 0.09140122127040107, 0.24354158436710194, 0.043430843974697690822, 0.032817367385225116, 0.04972751488388094, 0.28326439844046736, -0.4326054250659417, 0.4784156675384671, 0.4465063189919065, -0.257536436645459, 0.24097344200188098, 0.18175570781656225, -0.4791180202027972276, -0.3182561551522539, -0.46304444636686115, -0.43061371270290116, -0.2120348941287743, -0.0334574686865875, 0.1960086740778728, 0.4638202375045033, -0.2152093498131974, 0.2350239030472301, 0.2082756763456829, 0.47410451846892887, -0.3009128619991057, -0.04186301084599231, -0.34308543253502466, 0.4420996880013064, 0.050889645162975894, 0.267697827026101, 0.3025427569333691, 0.3955349019001252, 0.085298866332741465, 0.16816057424378272, 0.007607659553250379, -0.1779264210414363, 0.42647157121349033, 0.13665627144707548, -0.34324362696942745, 0.11429920209848454, -0.08207959911830542, -0.26418212874780755, -0.43894833288059876, -0.4952195970063201, -0.42942286949141306, 0.22141088099177642, 0.04339000419353434, 0.43702279911995845, 0.23735990971423548, 0.34830484195134614, 0.34455161688238245, 0.15533598585708763, -0.47578022991413094, -0.05411156654967564, -0.4183995347518509, -0.03400222039422707, 0.2772515695529433, 0.1003469670489433, -0.37722332676611947, -0.07944622014780078, 0.1698437807673967, 0.052467810278046145, -0.4997100634471432, 0.23768978365354254, -0.012343782250454294, -0.1335457035582308, -0.46268215590456874, 0.39071533272833414, -0.32576091677694996, -0.16482130072290457, 0.11125815993802013, -0.09906825059193702, 0.001736885303389557, 0.17745189769704683, -0.12399041285496881, -0.2757538141253163, 0.053259405410558736, -0.36604000360390243, -0.13122784361009432, 0.03299504731792047, -0.09898571742920

736, 0.4539058023232203, 0.10973609580492427, 0.11821269187523908, 0.2643732412849503, -0.01719439773578
102, -0.2996787968515977, -0.30133711745592273, -0.37727082951824165, 0.10352409512985217, -0.3088634452
6305376, 0.051247768258232296, 0.18469409379192858, 0.3888833444134622, 0.4328236323216814, 0.0016510724
008809152, 0.15400107464745005, -0.35284182922126284, -0.19584428359033978, 0.309063018453421, -0.360096
4045283508, -0.08828399991125646, -0.23277654003506787, -0.05858729229880277, 0.07712215035608216, 0.211
5200863354293, -0.18687791594636327, 0.3454464322992983, 0.47659734625767103, -0.22616729208185105, -0.1
1287308009010777, -0.039789166693874334, 0.12451010387756789, 0.25047836370794174, -0.4929789661799353,
0.16404066253750915, -0.1148838114058528, -0.07239562998116478, -0.12438296355964007, -0.027866785541607
042, 0.04931498960725633, -0.03449825831156528, -0.19537422231026602, -0.20107372447972782, 0.2735597356
7331104, -0.12953371700438143, 0.25065051814318207, 0.16101043537912962, 0.06859566260611005, -0.0234288
5369727754, -0.07153728893903932, 0.3766967545617794, 0.28476561352961005, 0.21246008995830235, -0.41967
81678493009, 0.301254349222959, 0.4890012982663785, 0.24035187775550582, -0.3679393870240034, -0.4652783
7932464666, -0.1682314192484985, -0.13287013494643574, -0.13690392118776562, -0.368308920435631, -0.4368
1902519374527, 0.025726474633270713, -0.12563953131587124, -0.413015293734566, 0.23046838021599703, 0.21
723230858199816, -0.14958450723537542, 0.4800362881025041, 0.34569244590542947, -0.06171350405904352, 0.
3840993233821971, 0.4844281292933623, 0.006104695742829 68, 0.017760183050057554, -0.2740289953228797, -
0.13510674472323558, 0.1035981186774454, 0.4642801700117388, 0.07975049297775416, 0.34562388918828857,
0.29552905242665717, -0.024807273553959952, -0.4710967168188136, 0.41200243395057656, -0.484045049920986
96, -0.3723548133013461, -0.042031240516121615, 0.2855927859777013, -0.022574846244402202, -0.2067643326
6210836, -0.44169745126279814, 0.13767328934905543, 0.4323588463107022, 0.27361066295491454, -0.23769565
906903556, -0.13903497536314458, 0.4060873070685498, -0.08938142567720353, 0.15434284045393587, 0.420733
41097182015, -0.37560028909690546, -0.161188638858912, 0.13985210076085175, 0.4665529707605769, 0.478643
0018568333, -0.07899402035014524, -0.4581780195827415, 0.09559943624907741, -0.07693364459434793, 0.3804
433344813496, 0.05608447174720388, -0.38890716968675854, -0.37715269353283454, -0.24566411982440584, 0.2
9253158907649857, -0.45640579809366455, 0.15050146370359863, -0.3763963370145549, 0.24902793312117544, -
0.30375551680356605, 0.12028345696496756, -0.41347885507672244, -0.15584087214687548, 0.2753473265677006
7, 0.17158608597269276, -0.24138733295485282, -0.27772913616533346, 0.1855761379363463, 0.44319139907795
124, -0.4510466556353381, -0.49545915288051157, 0.07260751212795347, -0.4145461910225856, 0.262013883276
52364, -0.2004917622975212, -0.3546431720236817, -0.2539362565834401, 0.04654442050857699, -0.4012377726
8994876, -0.13637950691559975, -0.28114683777937, 0.03750205097164294, 0.29119030801640633, -0.318653638
07414315, 0.08545488443561611, 0.3283738448556992, -0.4128002581002823, -0.367334735975252, 0.3564668621
110646, -0.3085016111016954, 0.003181467941493543, 0.24250934299169502, 0.3965632561476685, 0.3880038887
4166014, 0.20203194706544791, -0.3824267699153774, 0.1493560348454458, 0.027291741165895944, -0.43316697
941470284, -0.3613549639142676, 0.10405163203653067, 0.405293437657848, -0.16259547644423733, 0.07309309
444353695, 0.3886594171726998, -0.4599738807581666, -0.14464953950678905, 0.32092606797804235, -0.015999
6948808927, -0.3068478816842771, 0.4850394920851313, -0.20243016408586, -0.24805343828206794, -0.3510121
609809367, 0.188756125158842, 0.16793513958438455, 0.14389172880592904, -0.34534459438547227, 0.02415032
338272305, 0.1509265728072785, 0.19586661897712032, -0.017040845118141523, 0.2587757377649047, -0.047825
57709560331, 0.4332661572865866, -0.30181552385862853, 0.18973558065974105, 0.2908211270104837, 0.422924
95878076586, 0.38470586428426135, -0.2067763207790646, 0.09345988141848716, 0.05212027153248233, -0.3878
0708791916885, 0.44947299081666925, 0.401777846461417, -0.19934082188701863, -0.29681600217682214, 0.188
819116350046, 0.30097682309171625, -0.313536025020662, 0.08994173609041622, -0.30180299055676985, -0.351
5511940078766, 0.2726543660358518, 0.09615005152087985, -0.239159828673705, 0.02103883394228956, -0.1771
25253975579, -0.25016167076852713, 0.09711469240726245, -0.17950633523819481, 0.43147209604668935, -0.28
767010616833066, -0.50713673832802, 0.12135633388620348, 0.05809602139930237, 0.3549484148526745, 0.1694
7887028870146, -0.45328747691522475, 0.0006930296665986768, 0.2108607172966308, 0.31483341697601774, -0.
3917025090183145, 0.13236016064733747, 0.2704141925502791, 0.2983831399698569, 0.28906048416138386, 0.44
46500266239132, -0.05247915796093483, 0.18853292168231695, -0.30034664585957216, -0.3714304900747546, -
0.3074608407491197, -0.23429588099638032, -0.060630346175342575, 0.21713292864627654, 0.1271375292835059
7, -0.028595352435474064, 0.18261431426160224, -0.0735125011740827, -0.22182563956327583, -0.02193810609
415603, 0.1847118868751697, 0.3006658725332396, -0.3695765703736451, 0.08983969965264809, 0.133052587156
9509, -0.008457573154217912, -0.07746471571394065, 0.40762253852701535, 0.26836262233014097, -0.17857685
30119768, 0.0031388342796088153, -0.470587639346273, -0.0047017767881391, 0.2248478279750311, 0.48924246
34610377, -0.30674943414517164, -0.027916223609096402, 0.34521856112970684, -0.23434130120632635, -0.381
30705337742044, -0.33853227634965466, 0.45341517114768704, -0.14137804009280697, -0.3807249619942408, 0.
3072858347194305, 0.07291736551105477, 0.24199324815896583, -0.20434884779150053, 0.3370571960298263, -
0.41081622340143187, 0.39174771538724285, 0.05900251049761551, -0.3945844374527502, 0.3129547151852391,
-0.05494270832682813, -0.12457064187778544, -0.17551157219708335, 0.3070199097322912, -0.137080650121650
83, 0.4324177900937328, 0.3175402386167172, 0.3091768854300152, -0.15138723564856105, 0.0348820594685955
7, -0.1488086273771102, 0.02400934767739138, 0.08754586042391188, -0.0249232949610374, -0.37742046710999
366, -0.3170090147683591, -0.10550985804372948, 0.41163371031226526, -0.3404151695685027, -0.33149769925
784933, 0.41728804423133803, 0.16953181707857512, 0.35258515351744124, 0.2532752651535173, -0.2189487281
34418]

```
1000-element Vector{Float64}:
  0.037127131801273694
 -0.3579116316050869
  0.4558194989445373
 -0.36069975121248155
  0.25279440864812697
 -0.4874824481803701
  0.4875370394219454
  0.4406193193986143
 -0.3894187894165013
  0.3902347729979515
  ⋮
 -0.10550985804372948
  0.41163371031226526
 -0.3404151695685027
 -0.33149769925784933
  0.41728804423133803
  0.16953181707857512
  0.35258515351744124
  0.2532752651535173
 -0.218948728134418
```

Problem 2 (18 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.

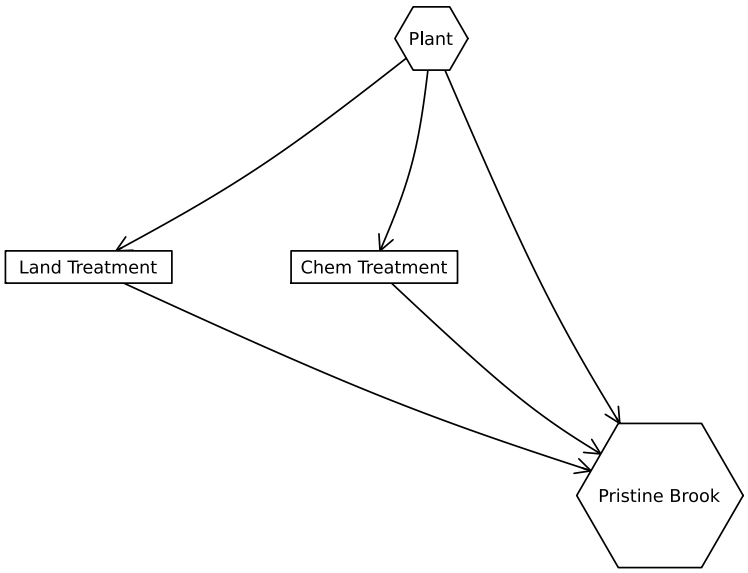
Cheap Plastic Products has analyzed two methods for reducing its discharges of YUK. Method 1 is land disposal, which costs $X_1^2/20$ dollars per day, where X_1 is the amount of wastewater disposed of on the land ($\mathrm{m}^3/\mathrm{day}$). With this method, 20% of the YUK applied to the land will 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 [5]:
```
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, markercolor=:w
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.

Answer 2.1

Designed of varaibles
X_1 = flow to land disposal (m^3/day)
X_2 = flow to chemical treatment (m^3/day)
X_3 = flow discharged directly to Pristine Brook (m^3/day)

Treatment Cost

Land disposal cost: $C_1(X_1) = \frac{X_1^2}{20}$
Chemical treatment cost: $C_2(X_2) = 1.5X_2$
Total daily cost: $C(X_1, X_2, X_3) = \frac{X_1^2}{20} + 1.5X_2$

The amount of YUK that will be discharged into Pristine Brook

Total plant outflow per day: $X_1 + X_2 + X_3 = 100$
Mass from land disposal: $D_1(X_1) = 0.2X_1$
Mass from chemical treatment: $D_2(X_2) = 0.005X_2^2$
Total YUK discharged into Pristine Brook: $Total = 0.2X_1 + 0.005X_2^2 + X_3$
Constraint: $0.2X_1 + 0.005X_2^2 + X_3 <= 20$

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:

In [38]:
```
function yuk(x1, x2)
    x3 = 100-x1-x2
```

```
    return (x1^2/20+1.5*x2, 0.2*x1+0.005*x2^2+x3)
end

cost, discharge = yuk(80, 19)
@show cost;
@show discharge;
```

```
cost = 348.5
discharge = 18.805
```

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:

In [40]:
```
x = [1, 2, 3, 4, 5]
y = [6, 7, 8, 9, 10]

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

```
a = [9.05, 10.7, 12.45, 14.3, 16.25]
b = [93.38, 91.645, 89.92, 88.205, 86.5]
```

Answer 2.2

By setting values for x_1 and x_2, and using $x_3 = 100 - x_1 - x_2$, we can calculate the cost and discharge with the mathematical model.
When applying the model to **multiple pairs of x_1 and x_2**, we need to use broadcasting (add a . before the function) so that it is applied element-wise.

Problem 2.3

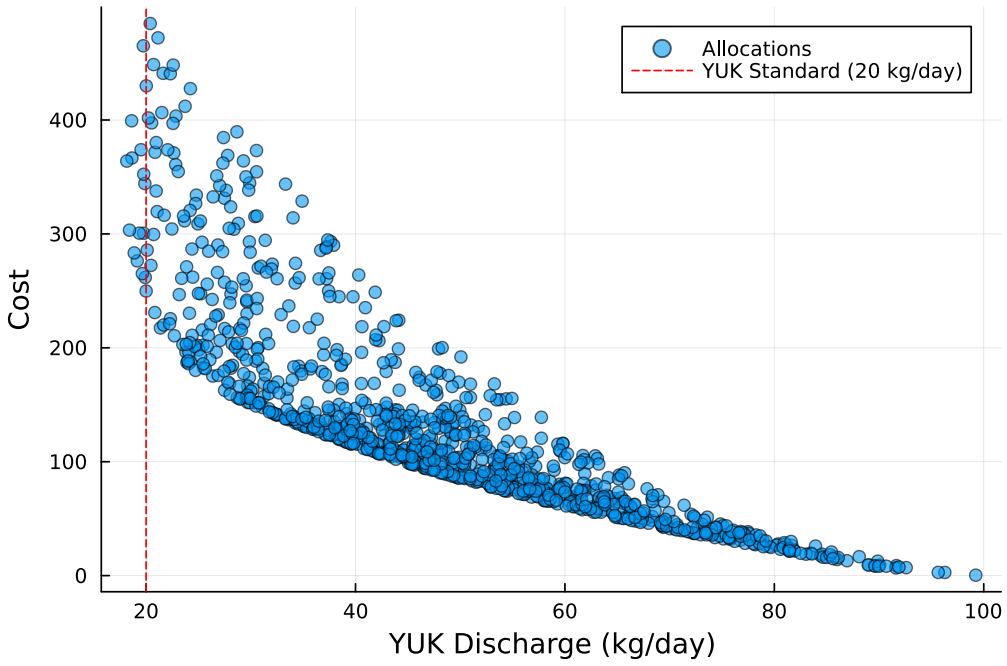
Use your function to experiment with 1,000 different combinations of wastewater discharge and treatment. You can do this with either a grid search or by sampling from a Dirichlet distribution (a $\mathrm{Dirichlet}(1, n)$ distribution will generate uniformly-weighted n-dimensional vectors whose components 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?

Answer 2.3

1. To test the mathematical model of the YUK treatment process with 1,000 random inputs, we **first generate 1,000 triplets (x_1, x_2, x_3) subject to the constraint $x_1 + x_2 + x_3 = 100$**. Using the **Dirichlet distribution together with the rand function**, we can sample 1,000 random allocations that satisfy this condition.
2. Then we put the 1000 random allocations into function $yuk(x_1, x_2)$ that we developed in Problem 2.2. The **results with 1000 paired input** are shown in the figure below.
3. Results that satisfy with the YUK effluent standard:
   The values **located to the left of the red line** represent allocations that satisfy the YUK effluent standard. Most of these solutions have **costs greater than 260 $/day**. The figure clearly shows that **higher costs are associated with lower YUK discharges**. To identify a better solution, we would need to generate more feasible allocations that meet the effluent standard and then determine the minimum possible cost among them.

In [48]:
```
# generate 1000 random values
n = 1000
d = Dirichlet(3, 1.0)  # make x1, x2, x3 sum to 1
allocations = rand(d, n)  # generate 1000 paired random values
allocations .*= 100   # make x1, x2, x3 sum to 100
```

```julia
# calculation of cost and discharge
results = [yuk(a[1], a[2]) for a in eachcol(allocations)]
cost    = map(first, results)  # first output from function yuk(x1,x2) is cost
discharge = map(last, results) # second output from function yuk(x1,x2) is discharge

# plot results
scatter(discharge, cost, label="Allocations", xlabel="YUK Discharge (kg/day)", ylabel="Cost", alpha=0.6)
vline!([20], linestyle=:dash, color=:red, label="YUK Standard (20 kg/day)")
```



In [ ]:

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)?

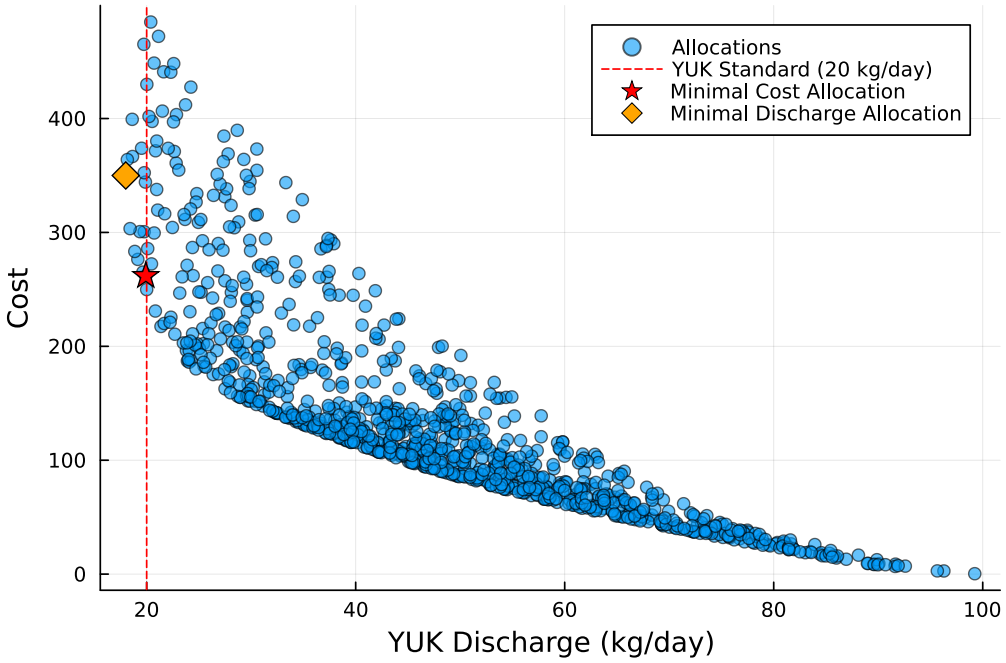
Answer 2.4

1. Cost minimize function
   We aimed to identify the allocation that minimizes cost while meeting the YUK effluent standard of 20 kg/day. Using findall, we filtered allocations (x_1, x_2, x_3) satisfying $discharge \leq 20$. We then applied argmin to find the allocation with the lowest cost.
   The results show that the minimal cost is 261.8 $/day, corresponding to a discharge of 19.92\ kg/day for the allocation$:x_1 = 64.97 kg/day, x_2 = 33.83 kg/day, x_3 = 1.2 kg/day$
2. Discharge minimize function
   To minimize discharge, we can set up $x3 = 0$, then $x2 = 100 - x3$,
   cost function will be $cost = 100 - 0.8x_1 - x_2 + 0.005x_2^2$,
   Whem $x_1 = 80$, $x_2 = 20$, $x_3 = 0$, the discharge reaches its minimum value of $18 kg/day$.
   The corresponding treatment cost is 350$/day.
3. I will adopt the **minimal cost strategy**. In the figure below, the red star represents the minimal cost allocation, while the orange marker indicates the minimal discharge allocation. It is evident that reducing the discharge below 20 kg/day **requires a disproportionately high cost for only a small additional decrease in YUK**

discharge. Considering **both treatment cost and discharge**, the minimal cost strategy provides the most balanced and efficient solution.

In [60]:
```
# cost minimize function
valid_indices = findall(d -> d <= 20, discharge)
valid_allocs = allocations[:, valid_indices]
valid_costs = cost[valid_indices]
valid_dis   = discharge[valid_indices]

# find minimal cost allocation
min_index = argmin(valid_costs)
min_alloc = valid_allocs[:, min_index]
cor_dis  = valid_dis[min_index]
min_cost = valid_costs[min_index]

println("x1, x2, x3: ", min_alloc)
println("Minimal cost: ", min_cost, "cor discharge: ",cor_dis)

# existing scatter of all allocations
scatter(discharge, cost, label="Allocations", xlabel="YUK Discharge (kg/day)", ylabel="Cost", alpha=0.6)
vline!([20], linestyle=:dash, color=:red, label="YUK Standard (20 kg/day)")

# add minimal cost allocation
scatter!([19.92], [261.8], color=:red, marker=:star5, label="Minimal Cost Allocation", markersize=8)

# add minimal discharge allocation
scatter!([18], [350], color=:orange, marker=:diamond, label="Minimal Discharge Allocation", markersize=8
```

```
x1, x2, x3: [64.97022592838785, 33.82635967849744, 1.2034143931147145]
Minimal cost: 261.7960523770342cor discharge: 19.91857262428767
```



References

Google