

# Rania — Waves Exam 3

April 29, 2025

TOTAL SCORE / 25

Comments and Scores for Each Problem Are on Last Page

This exam tests your fluency with the core of the Wolfram Language, as it was presented in *An Elementary Introduction to the Wolfram Language, 3rd Edition (EIWL3)*, Sections 25-34 and 38-41. There is one problem with two or three parts corresponding to each section. **Tip: all of them are meant to be quick. If you get bogged down, move on.**

## Directions:

After downloading this notebook, rename it with your first name in the filename. E.g., *Eli-Exam3.nb*, *Harper-Exam3.nb*, *Hexi-Exam3.nb*, *Jeremy-Exam3.nb*, *Rania-Exam3.nb*, *Tahm-Exam3.nb*, or *Walker-Exam3.nb*.

Then disconnect from the wifi and work the exam. Save your notebook early and often so that you don't lose work in progress.

**Your answers always go into the Wolfram Language Input cells that begin with a comment, e.g.,**

```
(* 1a *) foobar /@ Plus[Array]
```

**All your answers should execute and re-execute without warnings or error messages.**

You may refer to your downloaded copies of *EIWL3*, and anything else we developed in the course (like your cheat sheets!), but not to any web resources.

When you are done, save your notebook one last time, re-join the wifi, and then email it to me.

This exam was designed to require about 45 minutes, but if you need a full hour, that is ok. Everyone will stop at the one-hour mark.

## 1. Applying Functions (*EIWL3* Section 25)

(a)

Use **Map** with a *levelspec* to put a frame around each individual number in the array **Array[Plus, {10, 10}]** (we don't want frames around already-framed things — just one level of frames around the individual numbers).

In[103]:=

**(\* 1a \*)****Map[Framed, Array[Plus, {10, 10}], {2}]**

Out[103]=

```

{ {2, 3, 4, 5, 6, 7, 8, 9, 10, 11},
  {3, 4, 5, 6, 7, 8, 9, 10, 11, 12},
  {4, 5, 6, 7, 8, 9, 10, 11, 12, 13},
  {5, 6, 7, 8, 9, 10, 11, 12, 13, 14},
  {6, 7, 8, 9, 10, 11, 12, 13, 14, 15},
  {7, 8, 9, 10, 11, 12, 13, 14, 15, 16},
  {8, 9, 10, 11, 12, 13, 14, 15, 16, 17},
  {9, 10, 11, 12, 13, 14, 15, 16, 17, 18},
  {10, 11, 12, 13, 14, 15, 16, 17, 18, 19},
  {11, 12, 13, 14, 15, 16, 17, 18, 19, 20} }

```

**(b)**

Copy what you did in (a), but for this part, also turn the result into a grid using **Grid** and the “as an afterthought” syntax:

```
In[104]:=
(* 1b *)
Map[Framed, Array[Plus, {10, 10}], {2}] // Grid
```

```
Out[104]=
```

2	3	4	5	6	7	8	9	10	11
3	4	5	6	7	8	9	10	11	12
4	5	6	7	8	9	10	11	12	13
5	6	7	8	9	10	11	12	13	14
6	7	8	9	10	11	12	13	14	15
7	8	9	10	11	12	13	14	15	16
8	9	10	11	12	13	14	15	16	17
9	10	11	12	13	14	15	16	17	18
10	11	12	13	14	15	16	17	18	19
11	12	13	14	15	16	17	18	19	20

## 2. Pure Anonymous Functions (EIWL3 Section 26)

(a)

Use the **#** and **&** notation to create an anonymous function that cubes whatever is given it, and then use **/@** to apply it to every member of the list **{1, 2, 3, 4, 5}**.

```
In[105]:=
(* 2a *)
(#^3 &) /@ {1, 2, 3, 4, 5}
```

```
Out[105]=
{1, 8, 27, 64, 125}
```

(b)

Use the **#1**, **#2**, and **&** notation to create an anonymous function that divides its first argument by its second argument. Combine this with **Apply** and a *levelspec* to apply the function to **{{1, 2}, {2, 3}, {3, 4}, {4, 5}}**. Once you have this right, you will get  $\{\frac{1}{2}, \frac{2}{3}, \frac{3}{4}, \frac{4}{5}\}$ .

In[106]:=

**(\* 2b \*)****Apply[#1 / #2 &, {{1, 2}, {2, 3}, {3, 4}, {4, 5}}, {1}]**

Out[106]=

$$\left\{ \frac{1}{2}, \frac{2}{3}, \frac{3}{4}, \frac{4}{5} \right\}$$

### 3. Applying Functions Repeatedly (EIWL3 Section 27)

(a)

Use **Nest** to apply **Factorial** twice to **{1,2,3,4}**. If you have this right, 620,448,401,733,239,439,360,000 will be one of the elements of your answer.

In[107]:=

**(\* 3a \*)****Nest[Factorial, {1, 2, 3, 4}, 2]**

Out[107]=

{1, 2, 720, 620 448 401 733 239 439 360 000}

(b)

Use **NestList** to apply **Factorial** three times to **{1,2,3}**, as well as showing the results of doing it 0, 1, and 2 times. If you have this right, you will have an insanely large result at the third step. Do not go any higher, or I do not know what will happen to your computer.

[illegible]

#### 4. Tests and Conditionals (EIWL3 Section 28)

(a)

Use **PrimeQ** and **/@** to generate a **True** or **False** list that is twenty elements long expressing which numbers in **Range[20]** are prime.

```
In[109]:=
  (* 4a *)
  PrimeQ[#] & /@ Range[20]

Out[109]:=
  {False, True, True, False, True, False, True, False, False, False,
    True, False, True, False, False, False, True, False, True, False}
```

(b)

Combine **PrimeQ** with **Select** to only list the numbers in **Range[20]** that are prime.

```
In[110]:=
(* 4b *)
Select[Range[20], PrimeQ]

Out[110]=
{2, 3, 5, 7, 11, 13, 17, 19}
```

## 5. More About Pure Functions (EIWL3 Section 29)

(a)

Accomplish exactly the same thing as **Table[n\*(n-1)/2, {n,6}]** using **Array** and a pure function.

```
In[111]:=
(* 5a *)
Table[n*(n-1)/2, {n, 6}]
Array[#*(#-1)/2 &, 6]

Out[111]=
{0, 1, 3, 6, 10, 15}

Out[112]=
{0, 1, 3, 6, 10, 15}
```

(b)

Make some modifications to **FoldList[Plus, {1,2,3,4,5}]** so that it produces a list of the first 10 factorials. Instead of hand-coding the list up to 10, begin by first changing **{1,2,3,4,5}** to **Range[10]**.

```
In[113]:=
(* 5b *)
FoldList[Plus, {1,2,3,4,5}]
FoldList[Times, Range[10]]
(*just to check here
Factorial[Range[10]]*)

Out[113]=
{1, 3, 6, 10, 15}

Out[114]=
{1, 2, 6, 24, 120, 720, 5040, 40320, 362880, 3628800}
```

## 6. Rearranging Lists (EIWL3 Section 30)

(a)

Use **Transpose** and one of the *levelspec* options to turn

`{{{1,uno},{2,dos},{3,tres}},{{4,cuatro},{5,cinco},{6,seis}}}` into  
`{{{1,2,3},{uno,dos,tres}},{{4,5,6},{cuatro,cinco,seis}}}`

In[115]:=

```
(* 6a *)
```

```
Transpose[
```

```
{{{1, uno}, {2, dos}, {3, tres}}, {{4, cuatro}, {5, cinco}, {6, seis}}}, {1, 3, 2}]
```

Out[115]=

```
{{{1, 2, 3}, {uno, dos, tres}}, {{4, 5, 6}, {cuatro, cinco, seis}}}
```

(b)

Use **Flatten** and a *levelspec* option to turn

`{{{1,uno},{2,dos},{3,tres}},{{4,cuatro},{5,cinco},{6,seis}}}` into  
`{{1,uno},{2,dos},{3,tres},{4,cuatro},{5,cinco},{6,seis}}`

In[116]:=

```
(* 6b *)
```

```
Flatten[{{{1, uno}, {2, dos}, {3, tres}}, {{4, cuatro}, {5, cinco}, {6, seis}}}, 1]
```

Out[116]=

```
{{1, uno}, {2, dos}, {3, tres}, {4, cuatro}, {5, cinco}, {6, seis}}
```

## 7. Parts of Lists (EIWL3 Section 31)

(a)

Use the magical **All** position (you will need to use **All** more than once) to turn

`{{{Eli, Lerner},{Harper,Yonago},{Hexi,Jin}},{{Jeremy,Choy},{Rania,Zaki},  
{,Tahm,Loyd},{Walker,Harris}}}` into  
`{{Eli,Harper,Hexi},{Jeremy,Rania,Tahm,Walker}}`

In[117]:=

```
(* 7a *)
```

```
#[[All, All, 1] &@{{{Eli, Lerner}, {Harper, Yonago}, {Hexi, Jin}},
```

```
{{Jeremy, Choy}, {Rania, Zaki}, {Tahm, Loyd}, {Walker, Harris}}}
```

Out[117]=

```
{{Eli, Harper, Hexi}, {Jeremy, Rania, Tahm, Walker}}
```

(b)

Use a magical *negative positional argument* to extract `{Jeremy,Rania,Tahm,Walker}` from

`{{Eli,Harper,Hexi},{Jeremy,Rania,Tahm,Walker}}` and combine that with **Take** with a different magical *negative argument* to extract `{Tahm,Walker}`.

```
In[118]:=
(* 7b *)
Take[{{Eli, Harper, Hexi}, {Jeremy, Rania, Tahm, Walker}}][[-1]], -2]

Out[118]=
{Tahm, Walker}
```

## 8. Patterns (EIWL3 Section 32)

(a)

Use **Cases** to choose the lists that begin and end with the same letter in this list of lists (but look ahead to part (b) before you solve part (a)):

```
{
  {"a", "l", "u", "l", "a"},
  {"a", "l", "o", "h", "a"},
  {"a", "r", "a", "r", "a"},
  {"b", "o", "n", "u", "s"},
  {"c", "i", "v", "i", "c"},
  {"d", "e", "b", "e", "d"},
  {"e", "l", "b", "o", "w"},
  {"z", "a"},
  {"z", "z"}
}
```

```
In[119]:=
(* 8a *)
(*cooler way to do both is x_/;x[[1]]==x[[-1]]*)
Cases[{{"a", "l", "u", "l", "a"}, {"a", "l", "o", "h", "a"}, {"a", "r", "a", "r", "a"},
  {"b", "o", "n", "u", "s"}, {"c", "i", "v", "i", "c"}, {"d", "e", "b", "e", "d"},
  {"e", "l", "b", "o", "w"}, {"z", "a"}, {"z", "z"}}, {x_, __, x_}]

Out[119]=
{{a, l, u, l, a}, {a, l, o, h, a}, {a, r, a, r, a}, {c, i, v, i, c}, {d, e, b, e, d}}
```

(b)

The pattern **BlankNullSequence** has the shorthand `___`. Use `___` to improve the pattern you used in Part (a) so that the two-letter list `{z, z}` is also included in your result.



In[120]:=

```
(* 8b *)
Cases[{{"a", "l", "u", "l", "a"}, {"a", "l", "o", "h", "a"}, {"a", "r", "a", "r", "a"},
      {"b", "o", "n", "u", "s"}, {"c", "i", "v", "i", "c"}, {"d", "e", "b", "e", "d"},
      {"e", "l", "b", "o", "w"}, {"z", "a"}, {"z", "z"}}, {x_, ___, x_}]
```

Out[120]=

```
{{a, l, u, l, a}, {a, l, o, h, a}, {a, r, a, r, a}, {c, i, v, i, c}, {d, e, b, e, d}, {z, z}}
```

## 9. Assigning Names to Things (EIWL3 Section 38)

(a)

Use **Module** to compute  $x = \text{Factorial}[10]$ , and then produce  $\{x, x^2, x^3\}$ .

In[121]:=

```
(* 9a *)
Module[{x = Factorial[10]}, {x, x^2, x^3}]
```

Out[121]=

```
{3 628 800, 13 168 189 440 000, 47 784 725 839 872 000 000}
```

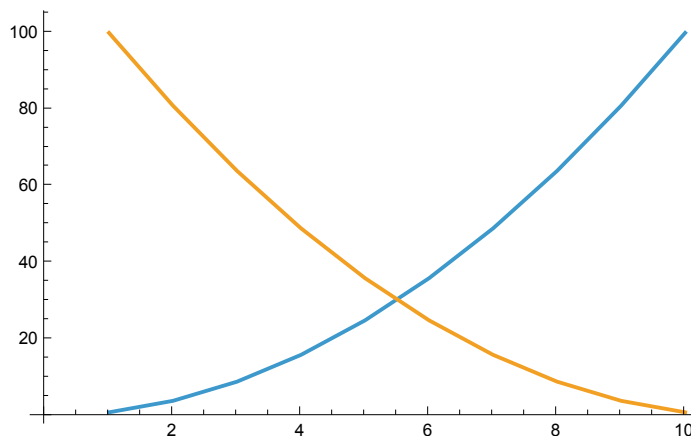
(b)

Inside **Module**, let  $\text{rangeSquared} = \text{Range}[10]^2$ , and then produce a list line plot of  $\text{rangeSquared}$  joined with  $\text{Reverse}[\text{rangeSquared}]$ .

In[122]:=

```
(* 9b *)
Module[{rangeSquared = Range[10]^2},
  ListLinePlot[{rangeSquared, Reverse[rangeSquared]}]
]
```

Out[122]=



## 10. Immediate and Delayed Values (EIWL3 Section 39)

(a)

Make a *one-character change* to this expression,

**Module**[**{x:=RandomInteger[10]}**,**{x,x<sup>2</sup>,x<sup>3</sup>,x<sup>4</sup>}**], so that it produces four different powers of the same random number instead of four different powers of different random numbers.

In[123]:=

```
Module[{x:=RandomInteger[10]}, {x, x2, x3, x4}]
(* 10a *)
Module[{x=RandomInteger[10]}, {x, x2, x3, x4}]
```

Out[123]=

```
{1, 9, 512, 625}
```

Out[124]=

```
{1, 1, 1, 1}
```

(b)

Make a *one-character change* to this expression,

**Module**[**{color=RandomColor[]}**,**Graphics**[**Table**[**Style**[**Disk**[**{i,0}**],**color**],**{i,5}**]]], so that it produces five different-color disks.

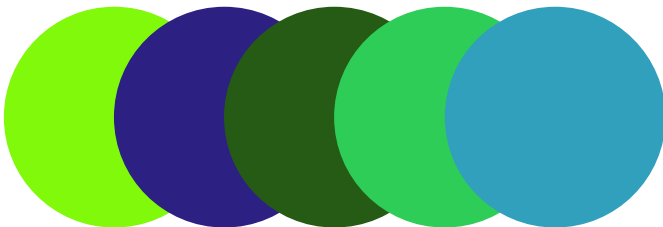
In[125]:=

```
Module[{color=RandomColor[]}, Graphics[Table[Style[Disk[{i,0}], color], {i,5}]]]
(* 10b *)
Module[{color:=RandomColor[]}, Graphics[Table[Style[Disk[{i,0}], color], {i,5}]]]
```

Out[125]=



Out[126]=



## 11. Defining Your Own Functions (EIWL3 Section 40)

(a)

Define a function **f** that takes a list of three elements and out of them makes a list of lists that contains all six possible orderings. Using **Permutations** will make this easy.

Include a test of your function as **f[1,2,3]** and make sure it gets **{{1,2,3},{1,3,2},{2,1,3},{2,3,1},{3,1,2},{3,2,1}}**.

In[127]:=

```
(* 11a *)
f[x_, y_, z_] := Permutations[{x, y, z}]
```

```
f[1, 2, 3]
```

Out[128]=

```
{{1, 2, 3}, {1, 3, 2}, {2, 1, 3}, {2, 3, 1}, {3, 1, 2}, {3, 2, 1}}
```

(b)

Define a function **g** that gives **1** for **g[0]**, and gives **n\*g[n-1]** for any integer **n** greater than **0**, *but don't use an If statement!* Include a test of your function as **g[6]** and make sure it gets **720**.

In[129]:=

```
(* 11b *)

(*it took me a hot minute but this is just factorials!*)
g[x_Integer] := x!
g[0]
```

```
g[6]
(*6*g[5]*)
```

```
(*g[100]
100*g[99]*)
```

Out[130]=

```
1
```

Out[131]=

```
720
```

## 12. More About Patterns (EIWL3 Section 41):

(a)

Use the replacement rule notation — e.g., **/.** and **->** — to exchange the first and last element in any list

containing two or more elements and test your replacement using the list `{alpha, beta, gamma, delta, epsilon}`.

```
In[132]:=
(* 12a *)
{alpha, beta, gamma, delta, epsilon} /.
  {a_, middleStuff___, c_} -> {a -> c, middleStuff, c -> a}
(*{a, b, c} /. a -> x
  {alpha, beta, gamma, delta, epsilon }*)

Out[132]=
{alpha -> epsilon, beta, gamma, delta, epsilon -> alpha}
```

(b)

Starting with `Characters/@RomanNumeral[Range[100]]`, select all the sequences corresponding to the Roman numerals that have XXX in them. Only wants ones with three consecutive XXs

```
In[133]:=
(* 12b *)

Cases[Characters /@ RomanNumeral[Range[100]], {___, "X", "X", "X", ___}]

Out[133]=
{{X, X, X}, {X, X, X, I}, {X, X, X, I, I}, {X, X, X, I, I, I}, {X, X, X, I, V},
 {X, X, X, V}, {X, X, X, V, I}, {X, X, X, V, I, I}, {X, X, X, V, I, I, I},
 {X, X, X, I, X}, {L, X, X, X}, {L, X, X, X, I}, {L, X, X, X, I, I},
 {L, X, X, X, I, I, I}, {L, X, X, X, I, V}, {L, X, X, X, V}, {L, X, X, X, V, I},
 {L, X, X, X, V, I, I}, {L, X, X, X, V, I, I, I}, {L, X, X, X, I, X}}
```

(c)

Use `StringJoin` to turn what you got in 12(b) into

`{XXX, XXXI, XXXII, XXXIII, XXXIV, XXXV, XXXVI, XXXVII, XXXVIII, XXXIX, LXXX, LXXXI, LXXXII, LXXXIII, LXXXIV, LXXXV, LXXXVI, LXXXVII, LXXXVIII, LXXXIX}`.

```
In[134]:=
(* 12c *)
StringJoin /@
  Cases[Characters /@ RomanNumeral[Range[100]], {___, "X", "X", "X", ___}]

Out[134]=
{XXX, XXXI, XXXII, XXXIII, XXXIV, XXXV, XXXVI, XXXVII, XXXVIII, XXXIX, LXXX,
 LXXXI, LXXXII, LXXXIII, LXXXIV, LXXXV, LXXXVI, LXXXVII, LXXXVIII, LXXXIX}
```

1. Applying Functions \_\_\_\_ / 2

2. Pure Anonymous Functions \_\_\_\_ / 2

3. Applying Functions Repeatedly \_\_\_\_ / 2

4. Tests and Conditionals \_\_\_\_ / 2

5. More About Pure Functions \_\_\_\_ / 2

6. Rearranging Lists \_\_\_\_ / 2

7. Parts of Lists \_\_\_\_ / 2

8. Patterns \_\_\_\_ / 2

9. Assigning Names to Things \_\_\_\_ / 2

10. Immediate and Delayed Values \_\_\_\_ / 2

11. Defining Your Own Functions \_\_\_\_ / 2

12. More About Patterns \_\_\_\_ / 3