Pr. 1.
Let $\boldsymbol{A}$ be a $M \times N$ matrix defined as: $A_{i,j} = i2^j$.

(a) Express $\boldsymbol{A}$ mathematically as an **outer product** of two appropriately defined vectors.

(b) Write (on paper) a one-line `Julia` expression for creating $\boldsymbol{A}$ when $M = 3$ and $N = 2$.

(c) Test your `Julia` expression (just for yourself, not graded).

Pr. 2.
Using the **determinant** properties: $\det(\boldsymbol{A}) = \det(\boldsymbol{A}^T)$ and $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{N\times N} \Rightarrow \det(\boldsymbol{AB}) = \det(\boldsymbol{A})\det(\boldsymbol{B})$,
solve the following problem: If $\boldsymbol{A}$ is **orthogonal**, what are the possible values of $\det(\boldsymbol{A})$?

Pr. 3.

(a) For $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$, prove that $\det(\boldsymbol{I} - \boldsymbol{xy}') = 1 - \boldsymbol{y}'\boldsymbol{x}$.

(b) Express $\det(\lambda\boldsymbol{I} - \boldsymbol{xy}')$ in terms of λ, $\boldsymbol{y}'\boldsymbol{x}$, and n.

(c) What are solutions of the equation $\det(\lambda\boldsymbol{I} - \boldsymbol{xy}') = 0$? (These are exactly the **eigenvalues** of the matrix $\boldsymbol{xy}'$.)

(d) When are the eigenvalues of the matrix $\boldsymbol{xy}'$ all equal to 0 even when the matrix is not equal to zero?

Hint: Use the following properties:

- If $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{n\times n}$, then $\det(\boldsymbol{AB}) = \det(\boldsymbol{A})\det(\boldsymbol{B})$.
- If $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, then $\det(\boldsymbol{A}^{-1}) = \dfrac{1}{\det(\boldsymbol{A})}$.
- If $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is **invertible** and $\boldsymbol{D} \in \mathbb{R}^{m\times m}$, then $\det\left(\begin{bmatrix}\boldsymbol{A} & \boldsymbol{B}\\ \boldsymbol{C} & \boldsymbol{D}\end{bmatrix}\right) = \det(\boldsymbol{A})\det(\boldsymbol{D} - \boldsymbol{CA}^{-1}\boldsymbol{B})$.
- If $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{D} \in \mathbb{R}_m^{m\times m}$ is invertible, then $\det\left(\begin{bmatrix}\boldsymbol{A} & \boldsymbol{B}\\ \boldsymbol{C} & \boldsymbol{D}\end{bmatrix}\right) = \det(\boldsymbol{D})\det(\boldsymbol{A} - \boldsymbol{BD}^{-1}\boldsymbol{C})$.

Pr. 4.
For $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, the **trace** of $\boldsymbol{A}$, denoted by $\mathrm{Tr}(\boldsymbol{A})$, is defined as the sum of its diagonal elements: $\mathrm{Tr}(\boldsymbol{A}) = \sum_{i=1}^n a_{ii}$.

(a) Show that the trace is a linear function; i.e., if $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{F}^{N\times N}$ and $\alpha, \beta \in \mathbb{F}$, then
$\mathrm{Tr}(\alpha\boldsymbol{A} + \beta\boldsymbol{B}) = \alpha\,\mathrm{Tr}(\boldsymbol{A}) + \beta\,\mathrm{Tr}(\boldsymbol{B})$.

(b) Show that $\mathrm{Tr}(\boldsymbol{AB}) = \mathrm{Tr}(\boldsymbol{BA})$, even though in general $\boldsymbol{AB} \neq \boldsymbol{BA}$, when both $\boldsymbol{AB}$ and $\boldsymbol{BA}$ are square.

(c) Let $\boldsymbol{S} \in \mathbb{R}^{N\times N}$ be **skew-symmetric**, i.e., $\boldsymbol{S}^T = -\boldsymbol{S}$. Show that $\mathrm{Tr}(\boldsymbol{S}) = 0$.

(d) Prove the converse of the previous part, or provide a counterexample.

(e) A matrix $\boldsymbol{A} \in \mathbb{F}^{N\times N}$ is called **idempotent** iff $\boldsymbol{A}^2 = \boldsymbol{A}$.
Show that the matrix $\boldsymbol{A} = \frac{1}{2}\begin{bmatrix}2\cos^2(\theta) & \sin(2\theta)\\ \sin(2\theta) & 2\sin^2(\theta)\end{bmatrix}$ is idempotent for all θ.

Pr. 5.
Let $\boldsymbol{U}_1, \boldsymbol{U}_2, \ldots, \boldsymbol{U}_k \in \mathbb{R}^{n\times n}$ denote **unitary** matrices. Show that the product $\boldsymbol{U}_1\boldsymbol{U}_2\cdots\boldsymbol{U}_k$ is a unitary matrix.

Pr. 6.

(a) Determine the **eigenvalues** (show your hand calculations) of $\boldsymbol{A} = \begin{bmatrix} 6 & 16 \\ -1 & -4 \end{bmatrix}$.

(b) For the matrix $\boldsymbol{A}$ above, compute the **determinant** and **trace** of $\boldsymbol{A}$.
Compute the product of the eigenvalues of $\boldsymbol{A}$ and the sum of the eigenvalues.
How do these compare to the determinant and the trace?

(c) Check your answers by invoking:

```
# in Julia
using LinearAlgebra
lambda, V = eigen(A)
```

Are the eigenvectors $\boldsymbol{V}$ orthogonal?
Display the matrix $\boldsymbol{V}'\boldsymbol{V}$ and submit a screenshot of the answer.

(d) (not graded)
Some programs sort the eigenvalues in a certain order. What order does `Julia` return for symmetric matrices?

Pr. 7.
Rewrite the expression

$$y = \sum_{i=1}^{n} \sum_{j=1}^{n} x_i^* A_{ij} x_j,$$

as a matrix-vector operation in terms of the $n \times n$ matrix $\boldsymbol{A}$ and the $n \times 1$ vector $\boldsymbol{x}$ whose ith entry is x_i.
Check your answer using `Julia` by trying some random examples.

Pr. 8.
Let $\boldsymbol{\Sigma}$ be an $m \times n$ diagonal matrix. Let the diagonal elements of $\boldsymbol{\Sigma}$ be denoted by $\sigma_1, \sigma_2, \ldots ....$ Let $\boldsymbol{U}$ and $\boldsymbol{V}$ be $m \times m$ and $n \times n$ matrices, respectively. Define $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}'$.

(a) Assume $m < n$: Express $\boldsymbol{A}$ as a sum of **outer product** matrices where each outer-product matrix is formed from the columns of $\boldsymbol{U}$ and $\boldsymbol{V}$. How many such outer-product matrices add up to form $\boldsymbol{A}$?

(b) Assume $m > n$: Express $\boldsymbol{A}$ as a sum of outer-product matrices where each outer-product matrix is formed from the columns of $\boldsymbol{U}$ and $\boldsymbol{V}$. How many such outer-product matrices add up to form $\boldsymbol{A}$?

(c) Generalize the answer above: what is the maximum number of outer-product matrices formed from the columns of $\boldsymbol{U}$ and $\boldsymbol{V}$, for $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}'$, that can add up to form $\boldsymbol{A}$?

Pr. 9.
The **Kronecker product** of two matrices $\boldsymbol{A} \in \mathbb{F}^{m_1 \times n_1}, \boldsymbol{B} \in \mathbb{F}^{m_2 \times n_2}$ is defined as follows:

$$\boldsymbol{A} \otimes \boldsymbol{B} = \begin{bmatrix} a_{11}\boldsymbol{B} & a_{12}\boldsymbol{B} & \ldots & a_{1n}\boldsymbol{B} \\ & & \ddots & \\ a_{m1}\boldsymbol{B} & a_{m2}\boldsymbol{B} & \ldots & a_{mn}\boldsymbol{B} \end{bmatrix} \in \mathbb{F}^{(m_1 m_2) \times (n_1 n_2)}.$$

Let $\text{vec}(\boldsymbol{C})$ denote the "vectorize" operation that returns as its output a $mn \times 1$ vector formed by stacking the n columns of $\boldsymbol{C} \in \mathbb{F}^{m \times n}$ on top of each other.
For arbitrary vectors $\boldsymbol{x} \in \mathbb{F}^M$ and $\boldsymbol{y} \in \mathbb{F}^N$, so that $\boldsymbol{x}\boldsymbol{y}^T \in \mathbb{F}^{M \times N}$, prove that

$$\text{vec}(\boldsymbol{x}\boldsymbol{y}^T) = \boldsymbol{y} \otimes \boldsymbol{x} \in \mathbb{F}^{MN}.$$

Verify (for yourself) by using commands `vec(x * y')` and `kron(y, x)` in `Julia` for some real vectors, and using `vec(x * transpose(y))` for some complex vectors.
(The equality above does not hold for $\text{vec}(\boldsymbol{x}\boldsymbol{y}')$ when $\boldsymbol{y}$ is complex.)

Pr. 10.
Discrete time **convolution** (a DSP term) of an input signal $x[n]$ and a filter $h[n]$ is defined in textbooks as:

$$y[n] = (h * x)[n] = (x * h)[n] \Rightarrow y[m] = \sum_{k=-\infty}^{\infty} h[k]x[m-k] = \sum_{n=-\infty}^{\infty} x[n]h[m-n],$$

where the "$*$" above denotes convolution, not ordinary multiplication.
In practice we often use finite-length signals and filters, and not all textbooks define this case clearly. The convolution of an input signal $(x[0], \ldots, x[N-1])$ with a finite impulse response (FIR) filter $(h[0], \ldots, h[K-1])$ becomes:

$$y[m] = \sum_{k=0}^{\min(K-1,m)} h[k]x[m-k] = \sum_{n=0}^{\min(N-1,m)} x[n]h[m-n], \quad m = 0, \ldots, M-1.$$

This is the type of convolution used in "convolutional neural networks" (a hot topic in machine learning).
Determine M, the length of the (possibly) non-zero part of $\boldsymbol{y}$, in terms of N and K.
We will use the notation $\boldsymbol{y}, \boldsymbol{h}, \boldsymbol{x}$ to denote the (finite) vectors of length M, K, and N, respectively.
Convolution is a linear operation, so we can express it as a **matrix-vector operation** of the form $\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x}$ where $\boldsymbol{H}$ is a matrix that you must determine and implement in this problem.
With some recycling of notation, you must implement a matrix $\boldsymbol{H}$ such that $\boldsymbol{y} = \boldsymbol{h} * \boldsymbol{x} = \boldsymbol{H}\boldsymbol{x}$
Caution: DSP notation uses signals that start at $n = 0$, whereas the first element of a vector in `Julia` (and `MATLAB`) is indexed by 1. This is something you must get used to handling properly.
Hint: It may be helpful to think through carefully the most reasonable size of $\boldsymbol{H}$ first. You should not augment $\boldsymbol{x}$, nor have any rows that are identically zero in $\boldsymbol{H}$.
Write a function called `convolution` that takes as its input the vectors $\boldsymbol{h}$ and $\boldsymbol{x}$ and returns as output both the matrix $\boldsymbol{H}$ and the column vector $\boldsymbol{y}$ (the convolved signal, implemented without `conv`).
Hint: Compare your answer with the builtin convolution function: `Julia`'s `conv` function (in the `DSP` package).
In `Julia`, your file should be named `convolution.jl` and should contain the following function:

```
"""
`H, y = convolution(h, x)`

Compute discrete convolution of the input vectors via
matrix multiplication, returning both the matrix `H` and result `y`

In:
- `h` vector of length `K`
- `x` vector of length `N`

Out:
- `H` `M x N` convolution matrix defined by `h`
- `y` vector of length `M` containing the discrete convolution of `h` and `x` computed using `H`.
"""
function convolution(h, x)
```

Submit your solution to the autograder by emailing it as an attachment to `eecs551@autograder.eecs.umich.edu`.

The material between the triple quotation marks `"""` is a `Julia` `docstring` and is the preferred way to document code. It supports markdown syntax. With this approach, you can type `?convolution` at the REPL or in a Jupyter notebook and see nicely formatted documentation. You are not required to include documentation when submitting code to the autograder, but making clear documentation is important in practice.

Autograder instructions:
Throughout the semester, all coding problems will be graded via an automated system that verifies the correctness of your code by evaluating it on a suite of test cases. To submit your code, simply attach your file to an email (from your `@umich.edu` account) and send it to: `eecs551@autograder.eecs.umich.edu`. (Later we may also provide another submission option in Jupyter.) The system will send an email response within a few seconds saying whether your code passes or fails its suite of (typically randomly generated) test cases.
If your code fails: Edit and resubmit your code via the same process. You have unlimited retries.
It is much more intelligent to first write your own test cases rather than trying to use the autograder as a debugger!
Once your code passes: You are done! You'll receive full credit for the problem. Do not submit anything to gradescope for this problem! The system already saved an electronic copy for our records. The "pass" email also

contains a confirmation code. If we accidentally fail to give you credit for the problem, forward to us the confirmation code and we'll record your credit. Points will be posted on Canvas at the end of the term.

Pr. 11.
To see how to use the convolution matrix from the previous problem in an image processing example, see:
`https://web.eecs.umich.edu/~fessler/course/551/julia/demo/hw1_show_conv_2d.html`
`https://web.eecs.umich.edu/~fessler/course/551/julia/demo/hw1_show_conv_2d.ipynb`
Modify that notebook to call your own `convolution` function and see how it works on the 2D image in that notebook.
To use this notebook, you will have to first install the `MIRT.jl` package by typing

```
] add MIRT
```

After installing, it can save time later to precompile it using

```
] precompile
```

Submit a screen shot of your filtered 2D image (the results of 2D convolution) to gradescope.
Your screenshot must include a figure title that shows your name or username.
Caution
When you submit scans of your work to gradescope, be sure to follow the instructions to properly associate submitted work to problems and subproblems. *Only work that is submitted properly will be graded!* For a class this size it is infeasible to make exceptions. Start the uploading process early to ensure you have time to submit properly.

Pr. 12.
Vectors $\boldsymbol{u}$, $\boldsymbol{v}$, $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{z}$ are all in $\mathbb{R}^N$ and we want to compute `z*u'*v*x'*y`

(a) Rewrite the code with parentheses to ensure the computation is as efficient as possible.

(b) Exactly how many multiplication operations are needed? Explain your answer.

Optional problem(s) below
(not graded, but solutions will be provided for self check; do not submit to gradescope)

Pr. 13.
Evaluate the following **block matrix** operations by simplifying the result to a single matrix (*i.e.*, no brackets) or scalar. (For simplicity, you may assume $\boldsymbol{A}, \boldsymbol{B}, \ldots, \boldsymbol{H}$ below are all square matrices of the same size. Most of the equalities generalize to rectangular matrices of appropriate sizes.) Below, $\boldsymbol{I}_2$ denotes the 2×2 identity matrix.

(a)
$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} & \boldsymbol{C} \end{bmatrix} \begin{bmatrix} \boldsymbol{D} \\ \boldsymbol{E} \\ \boldsymbol{F} \end{bmatrix} =$$

(b)
$$\begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{B} \end{bmatrix} \begin{bmatrix} \boldsymbol{C} & \boldsymbol{D} \end{bmatrix} =$$

(c)
$$\boldsymbol{A} \begin{bmatrix} \boldsymbol{B} & \boldsymbol{C} & \boldsymbol{D} \end{bmatrix} =$$

(d)
$$\begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{B} \\ \boldsymbol{C} \end{bmatrix} \boldsymbol{D} =$$

(e)
$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix} \begin{bmatrix} \boldsymbol{E} & \boldsymbol{F} \\ \boldsymbol{G} & \boldsymbol{H} \end{bmatrix} =$$

(f)
$$(\boldsymbol{I}_2 \otimes \boldsymbol{A}) \begin{bmatrix} \boldsymbol{E} & \boldsymbol{F} \\ \boldsymbol{G} & \boldsymbol{H} \end{bmatrix} =$$

(g)
$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \end{bmatrix} \begin{bmatrix} \boldsymbol{C} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{D} \end{bmatrix} \begin{bmatrix} \boldsymbol{E} & \boldsymbol{F} \end{bmatrix}' =$$

(h)
$$\text{trace}\left(\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix} \right) =$$

(i)
$$\det\left(\begin{bmatrix} \boldsymbol{A} & \boldsymbol{0} \\ \boldsymbol{B} & \boldsymbol{C} \end{bmatrix} \right) =$$

(j)
$$\boldsymbol{A}\text{diag}\lambda_1, \ldots, \lambda_N =$$