1. The calculation to determine monthly payments on a fixed-rate loan is based on
the following mathematics. Let $L$ be the amount of the loan. Let $p_k$ be the
remaining balance of the loan, $k$-months after you borrowed the money. Let $M$
be the (constant) monthly payment, and let $R$ be the annual interest rate, which
will be applied monthly (for a 6.375% interest rate, define $R = 0.06375$). Assuming
you start paying back after having the money for one month, the equations which
govern this are

$$
p_0 = L
p_{k+1} = (1 + \frac{R}{12})p_k - M
$$

The goal is that after $N$ months, the amount you owe is 0. Therefore, the terminal
condition is

$$p_N = 0$$

For notational purposes, define $\alpha := 1 + \frac{R}{12}$. It is straightforward to derive that
the numbers $R$ (or $\alpha$), $M$, $L$ and $N$ are related by

$$M = \frac{\alpha^N(\alpha - 1)}{\alpha^N - 1}L$$

Obviously, if $N$ and $R$ are specified, then it is easy to solve for $L$ in terms of $M$,
or visa-versa. Moreover, it is easy to show that the right-side is monotonically
decreasing with $N$, and monotonically increasing with $\alpha$. This means you can
program a simple bisection search to find the correct $\alpha$ or $N$.

Write a program called `loancalc.m` which has two input arguments: a 4-by-1
double vector called `LRMN`. The $4 \times 1$ variable `LRMN` contains the quantities $L$, $R$,
$M$ and $N$, in order listed. Actually, only 3 of the entries should contain actual
numbers, and one of the entries should be `NaN`. The entry containing `NaN` represents
the quantity to be computed. For example, if `LRMN` is `[30000 0.06 NaN 36]`,
then we are trying to compute the Monthly Payment, from the other 3 quantities
(principal = 30000, interest rate = 0.06, duration = 36 months). `loancalc` should
return the computed number.

2. Write a simple GUI that does loan calculations as demonstrated in class. Call the
program `loangui`, and the default call has no input arguments.

The tool should have 4 `uicontrols` and 4 mutually exclusive (one and only one
can be checked at any time) checkboxes (associated with each editable text).

The checked-box indicates which quantity is being computed from the other 3.
Initially, the Monthly Payment should have it’s check-box checked. Moreover, the
user should not be able to change the monthly payment. Once they fill in the other
3 boxes (which could be popup menus, or editable text, or a combination of both –
for instance, typical loan durations are 36 months, 60 months, 180 months and
360 months - those default values could be in a popupmenu, along with a Custom choice, which allows a user to type in an a number in a editable text uicontrol), the monthly payment immediately gets computed, and displayed, and a graph showing the amount owed and the cumulative amount paid as a function of the month number, from 0 to \( N \).

Any other uicontrol’s checkbox can be checked, which you must then make sure unchecks the currently checked box. That quantity now becomes the dependent variable - changing any of the other 3 should cause the graph and the dependent variable to be updated. While checked, protect the dependent variable from being changed, using, for example, the ‘enable’, ‘off’ property/value pair.

Alternatively, rather than four checkboxes, you can use a popupmenu to make the selection of what gets computed.

Finally, when \( N \) is the dependent variable, it is acceptable that it take on non-integer values (like, “the loan is paid off in 34.72 months.”). There is no need to use menus, but you can if you like. Otherwise, I am leaving the assignment vague.

3. Combine the betterkeypress.m and realtime.m examples from class into a new GUI. Call the file simpletetris.m. Here are the basic ideas.

- There should be an axes with a 10-by-10 array of patches.
- There should be a StartFalling and StopFalling button below or to the side of the axes.
- There should be a title, and the fall-rate should be displayed (nominal is 2 spaces/second, which is 0.5 seconds between falls).
- When the GUI gets created, the top-left patch should be visible, all others not visible. The StopFalling button should be disabled.
- Pressing the left-arrow and right-arrow keys “move” the patch to the left and right, immediately upon keypress (whether falling or not).
- The up-arrow and down-arrow buttons should alter the fall-rate by 0.2 spaces/second. The range of the allowable fall rates is between 0.4 spaces/second and 8 spaces/second. This should work whether falling or not.
- The patches start and stop falling in response to the button presses.
- When a patch “falls” off the bottom, it moves over (to the right) one column, and appears at the top. When a patch is in the right-most column, and is “moved” to the right, it ends up in the left-most column. When a patch is in the left-most column, and is “moved” to the left, it ends up in the right-most column.