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, for instance, program a simple bisection search to find the correct $\alpha$ or $N$. Other techniques (analytical manipulations, etc) are also possible approaches.

If you choose to use bisection, there are some simple bounds that hold which can be useful to implement. They are:

$$M > \frac{L}{N}, M > \frac{LR}{12}, M < L(1 + R/12), N < \frac{L}{M - LR/12}$$

Write a program called `loancalc.m` which has one input argument: 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 $[30000 \enspace 0.06 \enspace \text{NaN} \enspace 36]$, 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 one output argument, which is the computed value (in this case, the Monthly Payment).

2. Write a new class, `loanclass`. Inherit the `ReferAssignGetSet` class to handle the application-independent properties of referencing, assignment, get and set.
Use builtin parenthesis reference and assignment, and do not write horzcat, vertcat or cat, so that stacking works in the usual struct manner.

You do not need to write the following methods: char, display.

Your class should have the following public properties

- **Principal**: initial loan amount
- **InterestRate**: annual interest rate, 6.0 means 6%
- **MonthlyPayment**: monthly payment
- **Duration**: duration of loan, in months
- **DerivedProperty**: which (of above 4) is computed from the others
- **Balance**: See below
- **Equity**: See below

Some facts about the properties

(a) All properties are gettable.

(b) The **DerivedProperty** is always settable. Its possible values are the character strings **Principal**, **InterestRate**, **MonthlyPayment**, and **Duration**. The user need only type part of the string, and need not use upper case. Hence if A is a loanclass object, the user can type

```
>> A.DerivedProperty = 'm' % or 'mont' or 'Mo', etc
```

to change the property to MonthlyPayment.

(c) The property described by the value of **DerivedProperty** is not settable (it is the derived property). The other 3 properties are settable.

(d) The property described by the value of **DerivedProperty** is always updated to be the correct value, based on the calculation provided by loancalc.m.

(e) The syntax for the constructor should be

```
L = loanclass(DerivedProperty)
```

where DerivedProperty is one of the 4 possible values. Use default values (note that these are consistent) of Principal = 1, InterestRate = 0, MonthlyPayment = 1 and Duration = 1.

(f) With a final call to set, the constructor should handle

```
L = loanclass(DerivedProperty,SetableProp1,Value1,SetableProp2,Value2)
```

For example,

```
L = loanclass('Monthly','Prin',30000,'Rate',9,'Durat',36)
```

will create a loanclass object with a principal of 30,000, interest rate of 9%, 3-year (36 month) duration. The MonthlyPayment is derived from these 3 quantities.

(g) The **Balance** property is only gettable. It is a column vector, of length **Duration**. The k’th entry is the balance after the k’th month. Using the
notation from the previous assignment, this is simply $p_k$. The last entry of $Balance$ should be 0.

(h) The $Equity$ property is only gettable. It is a column vector, of length $Duration$. The $k$'th entry is the equity after the $k$’th month. Using the notation from above, this is

$$E_k = L - p_k$$

The last entry of $Equity$ should be the loan amount.