SM-2302 Labs (R2)

1. Represent the following JSON data as a list in R.

```
{
  "firstName": "John",
  "lastName": "Smith",
  "age": 25,
  "address":
  {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": 10021
  },
  "phoneNumber":
  [
    {
      "type": "home",
      "number": "212 555-1239"
    },
    {
      "type": "fax",
      "number": "646 555-4567"
    }
  ]
}
```

2. The operator `%*%` performs matrix multiplications (see the help file for more info). Use R to determine the matrix product of the following two matrices:

$$A = \begin{pmatrix} 7 & 8 & 2 \\ 5 & 9 & 1 \\ 4 & 6 & 3 \end{pmatrix} \qquad \text{and} \qquad B = \begin{pmatrix} 13 & -17 \\ 0 & 19 \\ -3 & -5 \end{pmatrix}$$

3. Read about the function `outer()` in the help file. Then, recreate the following patterned matrix. *Hint: You might need to think about the (mathematical) function modulo, which returns the remainder after division. In R, this is achieved by* `%%`.

$$\begin{pmatrix} 0 & 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 \\ 1 & 0 & 8 & 7 & 6 & 5 & 4 & 3 & 2 \\ 2 & 1 & 0 & 8 & 7 & 6 & 5 & 4 & 3 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \end{pmatrix}$$

4. Below are 100 values less than 50 that are randomly generated. (Note that the `set.seed()` function specifies the starting seed value for the pseudo random number generator for replicability)

```
set.seed(123)
(x <- sample(1:50, size = 100, replace = TRUE))
```

```
##   [1] 31 15 14  3 42 50 43 37 14 25 26 27  5 27 28  9 29 35  8 26  7
```

```
##  [22] 42  9 19 36 14 17 43 39 12 15 32 42 45  7  9 41 10 23 27  7 27
##  [43] 32 38 25 34 29  5  8 12 13 18 33 27 25 38 21 15 41 47 26 31 16
##  [64] 30  6 43  8 22 22 39 31 48 17 50 49 34  4 13  5 25 22 25 32 46
##  [85] 25 23 35 40 48 30 12 31 46 30 35 14 29 32  7  3
```

Write down how you would create a subset to accomplish each of the following:

(a) Select every third value starting at position 2 in `x`.

(b) Remove all values with an odd index (e.g. 1, 3, etc.)

(c) Remove every 4th value, but only if it is odd.

5. Some data providers choose to encode missing values using values like `-999`. Below is a sample data frame with missing values encoded in this way.

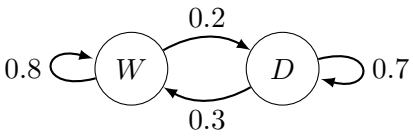
```
d <- data.frame(
  patient_id = c(1, 2, 3, 4, 5),
  age = c(32, 27, 56, 19, 65),
  bp = c(110, 100, 125, -999, -999),
  o2 = c(97, 95, -999, -999, 99)
)
```

(a) Using the subsetting tools we've discussed come up with code that will replace the `-999` values in the `bp` and `o2` column with actual `NA` values. Save this as `d_na`.

(b) Once you have created `d_na` come up with code that translate it back into the original data frame `d`, i.e. replace the `NA`s with `-999`.

6. A Markov chain (or Markov process) is a mathematical system that experiences transitions from one state to another according to certain probabilistic rules. Real-world applications are plentiful: queues in supermarkets, market trend predictions, auto-complete features, election polls, and so on.

Consider a simple Markovian weather system, whereby there are two states wet (W) and dry (D). The probabilities of transitioning between states are depicted in the diagram below.



For instance, if it is wet today, then tomorrow will be dry with probability 0.2 (i.e. $W \to D$ with probability 0.2). Transitioning further on requires simply multiplying the probabilities together, e.g. $\Pr(W \to D \to W) = 0.2 \times 0.7 = 0.14$.

Let W_i and D_i are the events that the weather is wet and dry on day i respectively. Suppose that today (day 1) is wet. The probability that it is also wet on day 3 is calculated as

$$\begin{aligned}\Pr(W_3) &= \Pr(W_1 \to W_2 \to W_3 \text{ or } W_1 \to D_2 \to W_3) \\ &= \Pr(W_1 \to W_2 \to W_3) + \Pr(W_1 \to D_2 \to W_3) \\ &= 0.8 \times 0.8 + 0.2 \times 0.3 = 0.7\end{aligned}$$

In a similar fashion,

$$\Pr(D_3) = 0.8 \times 0.2 + 0.2 \times 0.7 = 0.3$$

More generally, the following relationship holds:

$$\begin{bmatrix}\Pr(W_{i+1}) & \Pr(D_{i+1})\end{bmatrix} = \begin{bmatrix}\Pr(W_i) & \Pr(D_i)\end{bmatrix} \begin{bmatrix}0.8 & 0.2 \\ 0.3 & 0.7\end{bmatrix}$$

Thus, we can obtain the probability of the state W and D at any given time by recursively evaluating the relationship above. The square matrix at the end of the RHS above is called the *transition matrix*.

Write a function that returns the weather forecast for the next `n` days. Note that this forecast depends on the initial state of the weather (day 1), so your function should be able to adjust for this.