Name: *Student ID:*

CSC 120 (R Section) — Mid-term Test — 2015-02-24 — Answers

No books, notes, or calculators are allowed. You have 50 minutes to write this test. The eight questions are worth equal amounts.

Question 1: Suppose we type commands into the R console as shown below. (The `>` and `+` characters at the beginnings of the lines are the prompts printed by R, not part of what was typed.) In the six blank spaces below, fill in the output that R will produce as a result of these commands.

```
> 1+7*10

ANSWER: 71


> a <- 3
> (a+1)*a

ANSWER: 12


> b <- a + 2
> a <- a + 1
> a*b

ANSWER: 20


> x <- c(3,5,1)
> x - 1

ANSWER: 2  4  0


> x[2] <- 10
> x

ANSWER: 3  10  1


> y <- x
> y[1] <- 6
> x[1] + y[1]

ANSWER: 9
```

Question 2: Write an R funcion called `first_last_sum_prod` that takes two arguments that are numeric vectors, and returns the sum of the first and last elements of the first argument multiplied by the sum of the first and last elements of the second argument. For example, `first_last_sum_prod (c(4,9,6), c(2,1,8,3))` should return the value 50, which is 4+6 times $2 + 3$. Write your definition for `first_last_sum_prod` below:

```
first_last_sum_prod <- function (x,y)
    (x[1] + x[length(x)]) * (y[1] + y[length(y)])
```

Question 3: Write an R funcion called `nsame` that takes one argument, called `vec`, that you should assume is a numeric vector, and that returns the number of elements in `vec`, not including the first, that are the same as the element immediately before.

Here are two example calls of `nsame`:

```
> nsame (c (4,1,3,4,1))
[1] 0
> nsame (c (4,1,1,4,5,5,5,3))
[1] 3
```

Write your definition for `nsame` below:

```
nsame <- function (vec) {
    s <- 0
    for (i in 2:length(vec))
        if (vec[i] == vec[i-1]) s <- s+1
    s
}
```

Question 4: Write an R funcion called `X_matrix` that takes one argument, called `n`, that you should assume is a positive integer, and that returns an `n` by `n` matrix of numbers, which are zero except that they are one on the diagonal from the top left to bottom right and on the diagonal from the top right to the bottom left.

Here are two example calls of `X_matrx`:

```
> X_matrix(4)
     [,1] [,2] [,3] [,4]
[1,]    1    0    0    1
[2,]    0    1    1    0
[3,]    0    1    1    0
[4,]    1    0    0    1
> X_matrix(5)
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    0    0    0    1
[2,]    0    1    0    1    0
[3,]    0    0    1    0    0
[4,]    0    1    0    1    0
[5,]    1    0    0    0    1
```

Write your definition for `X_matrix` below:

```
X_matrix <- function (n) {
    M <- matrix (0, nrow=n, ncol=n)
    for (i in 1:n) {
        M[i,i] <- 1
        M[i,n-i+1] <- 1
    }
    M
}
```

Question 5: Consider the R function defined below:

```
mystery1 <- function (x,y) {
    a <- 0
    for (i in 1:length(x)) {
        if (x[i] <= y)
            if (x[i] >= -y)
                a <- a + 1;
    }
    a
}
```

For each of the calls of this function below, write down what value it will return:

```
> mystery1 (c(3,-2,5,6,4,-1,-9,0), 4)
```

ANSWER: 5

```
> mystery1 (rep(c(0,2,3),1000), 2.5)
```

ANSWER: 2000

```
> mystery1 (11:2000, 15)
```

ANSWER: 5

Below, describe in words what this function does in general, when given a numeric vector as its first argument, and a single non-negative number as its second argument.

It returns the number of elements in its first argument that have absolute value less than or equal to the second argument.

Question 6: Consider the R function defined below:

```
mystery2 <- function (x) {
    if (x < 1) stop("invalid argument")
    n <- 1
    m <- 10
    while (x >= m) {
        m <- 10 * m
        n <- n + 1
    }
    n
}
```

For each of the calls of this function below, write down what value it will return:

```
> mystery2(3.45)
```

ANSWER: 1

```
> mystery2(345)
```

ANSWER: 3

```
> mystery2(3456.7)
```

ANSWER: 4

Below, describe in words what this function does in general, when given a single number as its argument:

If the argument is less than one, the function stops with the message "invalid argument". Otherwise, it returns how many digits to the left of the decimal point there are in the decimal representation of its argument.

Question 7: Suppose we type commands into the R console as shown below. (The `>` and `+` characters at the beginnings of the lines are the prompts printed by R, not part of what was typed.) In the seven blank spaces below, fill in the output that R will produce as a result of these commands.

```
> y <- 9
> z <- 0
> whathappens <- function (x, y, z = 7) {
+   a <- 1000*x + 100*y + 10*z
+   a + y
+ }
> a <- 3
> whathappens (2, 8, 9)
```

ANSWER: 2898

```
> y
```

ANSWER: 9

```
> a
```

ANSWER: 3

```
> x <- 6
> whathappens (1, 2)
```

ANSWER: 1272

```
> whathappens (z, 1, 2)
```

ANSWER: 121

```
> z
```

ANSWER: 0

```
> whathappens (whathappens(0,0), 0, 0)
```

ANSWER: 70000

Question 8: Write an R funcion called `zero_locations` that takes one argument, called `M`, that is a matrix of numbers, and returns a list containing two-element vectors indicating the positions of all the zeros in `M`, with the two elements giving the row and the column index of a zero. The order of this list does not matter (you can make it have whatever order you like).

For example, here is call of this function and its output:

```
> A <- matrix (c(2,0,1,3,0,5,0,2), nrow=2, ncol=4)
> A
     [,1] [,2] [,3] [,4]
[1,]    2    1    0    0
[2,]    0    3    5    2
> zero_locations(A)
[[1]]
[1] 1 3

[[2]]
[1] 1 4

[[3]]
[1] 2 1
```

Note that `nrow(M)` gives the number of rows in the matrix `M`, and `ncol(M)` gives the number of columns.

Write your defintion for `zero_locations` below:

```
zero_locations <- function (M) {
    L <- list()
    n <- 0
    for (i in 1:nrow(M)) {
        for (j in 1:ncol(M)) {
            if (M[i,j] == 0) {
                n <- n + 1
                L[[n]] <- c(i,j)
            }
        }
    }
    L
}
```