Name: *Student ID:*

CSC 121 — Mid-term Test — 2017-03-03 — WITH ANSWERS

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

Question 1: Suppose you type commands into the R console as shown below. (The `>` 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.

```
> 10+4/2

[1] 12


> a <- 3
> b <- a + 2
> (a+1)*b-7

[1] 13


> a <- a - 5
> a*b

[1] -10


> ab <- c(10,15,5,30)
> ab/5

[1] 2 3 1 6


> ab[2] <- 0
> ab + a*b

[1]   0 -10  -5  20


> y <- ab
> y[3] <- 6
> c(ab,y)

[1] 10  0  5 30 10  0  6 30
```

Question 2: Write an R function called `shortest_first` that takes as arguments two vectors, and returns the vector combining in sequence the shorter of these vectors with the longer of these vectors. If the two vectors are the same length, the first should be combined with the second.

Here are three example calls of `shortest_first`:

```
  > shortest_first (c(3,9,1), c(2,0))
  [1] 2 0 3 9 1
  > shortest_first (c(3,9), c(2,0,1))
  [1] 3 9 2 0 1
  > shortest_first (c(4,9,1), c(7,4,8))
  [1] 4 9 1 7 4 8
```

Write your definition for `shortest_first` below:

```
shortest_first <- function (v1, v2)
    if (length(v2) < length(v1)) c(v2,v1) else c(v1,v2)
```

Question 3: Consider the R function defined below:

```
    mystery <- function (x,y) {
        for (j in 1:length(y)) {
            if (y[j] > x[length(x)])
                x <- c(x,y[j])
        }
        x
    }
```

Write down what will be printed by the R console if the commands below are done after the mystery function has been defined.

```
> mystery (c(4,2,5,6), c(5,10,9,12))

[1]  4  2  5  6 10 12


> mystery (c(90,95), 1:100)

[1]  90  95  96  97  98  99 100


> mystery (mystery(c(4,2),c(1,8)), mystery(c(2,6),c(7,1,10)))

[1]  4  2  8 10
```

Question 4: Write an R function called `around_dots` that takes two arguments, the first a single string, and the second a positive integer. (You don't need to check that the arguments are actually like this.) This function should return as its value a single string that has copies of its first argument at the beginning and end, separated by dots (`"."`), with the number of dots given by the second argument.

Here is an example call of this function (from the R console):

```
> around_dots ("dog", 5)
[1] "dog.....dog"
```

You should write this function using a `for` or `while` loop. Do not try to avoid a loop by using R features that achieve the effect of repetition some other way.

Write your definition for `around_dots` below:

```
around_dots <- function (s, n)
{
    r <- s

    for (i in 1:n)
        r <- paste(r,".",sep="")

    paste(r,s,sep="")
}
```

Question 5: Suppose we type commands into the R console as shown below. (The > 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.

```
> xyz <- 8
> funnyfun <- function (x, y=3, z=5) xyz + x * y * z
> funnyfun(3,2,-1)

[1] 2
```

```
> funnyfun(2)

[1] 38
```

```
> y <- 2
> funnyfun(2)

[1] 38
```

```
> funnyfun(y,xyz,1)

[1] 24
```

```
> x <- 4
> xyz <- 2
> funnyfun(x+1,y-1)

[1] 27
```

```
> funnyfun(0) + funnyfun(1)

[1] 19
```

Question 6: Write an R function called `find_maxima`, which takes one argument that is a matrix of numbers (you don't need to check that it is), and which returns a list of locations of local maxima in this matrix, with a location being represented by a vector of two numbers giving the row and column indexes where there is a local maximum. A local maximum is a place in the matrix that is not on an edge or in a corner where the number is greater than the numbers located above, below, left, and right. If there is more than one local maximum, they may appear in any order in the list that is returned.

Here is an example of what `find_maxima` should do:

```
> M
     [,1] [,2] [,3] [,4] [,5]
[1,]    5    4    2    0    1
[2,]    1    1    8    7    3
[3,]    9    7    3    9    4
[4,]    3    3    9    2    0
> find_maxima(M)
[[1]]
[1] 2 3

[[2]]
[1] 3 4
```

```
find_maxima <- function (M) {
    L <- list()
    if (nrow(M) > 2 && ncol(M) > 2) {
        for (i in 2:(nrow(M)-1)) {
            for (j in 2:(ncol(M)-1)) {
                if (M[i,j] > M[i-1,j] && M[i,j] > M[i+1,j]
                     && M[i,j] > M[i,j-1] && M[i,j] > M[i,j+1]) {
                    L[[length(L)+1]] <- c(i,j)
                }
            }
        }
    }
    L
}
```