Question 1. [6 marks]

The following commands are typed 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.
You do not have to add the `[1]` at the beginning of each output line.

```
2 5 2 5 2 5

0.0 2.5 5 7.5 10.0  (decimals on 0, 10 or not are fine)

FALSE

6

"csc207"

0.1680415 0.1680415 0.1680415
```

Question 2. [4 marks]

Consider the following function, `ListMaker`, which returns a dictionary (also known as a named list).

```
ListMaker <- function(v) {

  L <- list()
  for (i in 1:length(v)) {
    if (i != length(v)) {
      valueCharacter <- as.character(v[i])
      if (valueCharacter %in% names(L)) {
        L[[valueCharacter]] <- append(L[[valueCharacter]], v[i + 1])
      } else {
        L[[valueCharacter]] <- v[i + 1]
      }
    }
  }

  return(L)

}
```

Consider the dictionary returned by calling the function on the following vector argument. For the function call below, list each of the keys of the returned dictionary, and the values for each key. Make sure you clearly indicate which of the keys or values are strings by placing "quotes" around them.

```
> ListMaker(c(1, 3, 3, 5))
```

Key	Value
'1'	3
'3'	3 5

Question 3. [5 marks]

Complete the function body of the function `EdgeReplace` according to its docstring and the test cases below. Do not use any functions we haven't talked about in class.

```
EdgeReplace <- function(M, n, times) {

 ## Using a nested for loop ##
  for (i in 1:nrow(M)) {
    rowCount <- 0
    for (j in 1:ncol(M)) {
      if (M[i,j] == n) {
        rowCount <- rowCount + 1
      }
    }
    if (rowCount == times) {
      M[i, 1] <- 0
      M[i, ncol(M)] <- 0
    }
  }

  return(M)

}

  ################################

  ## Using one for loop ##
  for (i in 1:nrow(M)) {
    row <- M[i,]
    nElements <- row[row == n]

    if (length(nElements) == times) {
      M[i, 1] <- 0
      M[i, ncol(M)] <- 0
    }
  }

  return(M)

}
```

Question 4. [5 marks]

```
ChangeTransaction <- function(transactionID, change) {

  # Read the data from the file into a data frame and assign it to 'accountData'.
  accountData <- read.table("account.txt", header=TRUE)

  # Change accountData as needed.
  for (i in transactionID:nrow(accountData)) {
    if (i == transactionID) {
      accountData$amount[i] <- accountData$amount[i] + change
      # OR accountData[i,]$amount will also be accepted
    }
    accountData$balance[i] <- accountData$balance[i] + change

  }

  # Return the changed data frame.
  return(accountData)

}
```

Part B (1 mark)
Given a data frame called `accountData` that was read in from `account.txt`, write an R statement that plots a *line* graph (all other plot settings default) where the x-axis is the **transaction ID** and the y-axis is the **balance** at that transaction ID.
Remember that you are creating a new plot, not adding to a previous one.

Write your code in the box below.

```
plot(accountData$transactionID, accountData$balance, type="l")
```