The University of British Columbia

Irving K. Barber School of Sciences

DATA 101

Practice for the Midterm Break Week

1. The secant method for finding the solution x of an equation of the form

$$f(x) = 0$$

is

$$x_n = x_{n-1} - \frac{(x_{n-1} - x_{n-2})f(x_{n-1})}{f(x_{n-1} - f(x_{n-2})}$$

where two initial guesses x_1 and x_2 must be specified beforehand. With these guesses, we can use the formula to calculate x_3, and with x_2 and x_3, we calculate x_4, and so on. Typically, the solution is found to a few digits of accuracy in fewer than 5 steps, i.e. x_6 should be a good approximation to the solution.

(a) Write a function named `secant()` with three inputs including `x1`, `x2` and a function `f` which returns the approximate solution of $f(x) = 0$ based on 3 steps of the formula given above. (An example of `f` could be `cos` for which $x = 1.57$ is a good approximation to a solution of $\cos(x) = 0$.)

```
secant <- function(x1, x2, f) {
     fx1 <- f(x1)
     for (i in 1:4) {
         fx2 <- f(x2)
         x <- x2 - (x2-x1)*fx2/(fx2 - fx1)
         x1 <- x2; x2 <- x
         fx1 <- fx2
     }
 return(x)
 }
```

(b) Use the `secant` function just created to verify that a solution to $cos(x) = 0$ is $x = 1.57$. Use starting values $x_1 = 1.56$ and $x_2 = 1.58$.

```
secant(1.56, 1.58, cos)

## [1] 1.570796
```

(c) Write a function `f()` which takes a single argument `x` and returns the value of the function

$$f(x) = x^3 - 2x + 3.$$

Apply the `secant()` function to find the solution of

$$x^3 - 2x + 3 = 0.$$

Use -2 and -1.8 as your starting guesses.

```r
f <- function(x) {
   x^3 - 2*x + 3
}
```

```r
secant(-2, -1.8, f)

## [1] -1.893289
```

2. Finish writing the function below which should be called `WHunif` and which computes n uniform pseudorandom numbers on the interval $[0, 1]$ using a random number generator (called the Wichman-Hill generator):

   For $j = 1, 2, \ldots, n$,

$$
\begin{aligned}
x_j &= 171\ x_{j-1} \bmod 30269 \\
y_j &= 172\ x_{j-1} \bmod 30307 \\
z_j &= 170\ x_{j-1} \bmod 30323 \\
v_j &= x_j/30269 + y_j/30307 + z_j/30323. \\
u_j &= v_j - [v_j]
\end{aligned}
$$

   where x_0, y_0, and z_0 are all initial seeds, and $[v]$ is the integer part of v, or *floor* of v.

   Your function should take `n`, and the seeds `x`, `y`, `z` as arguments, and return the vector `u` as output.

   (This is a real generator which is actually used in the R function `runif()`).

```
? <- function(?, x, y, z) {
    u <- numeric(n)
    for (i in 1:n) {
    ?
    ?
    ?
    ?
    ? <- v - floor(v)
    }
?
}
```

```r
WHunif <- function(n, x, y, z) {
    u <- numeric(n)
    for (i in 1:n) {
    x <- (171*x)%%30269
    y <- (172*y)%%30307
    z <- (170*z)%%30323
    v <- x/30269 + y/30307 + z/30323
    u[i] <- v - floor(v)
    }
```

```
u
}
```

3. Obtain 20 uniform numbers using the above function with seeds 1, 2, and 3.

```
WHunif(20, 1, 2, 3)

##  [1] 0.03381877 0.77754189 0.05273525 0.74462407 0.49036219 0.98285437
##  [7] 0.80915099 0.71338138 0.80102091 0.98958603 0.91685632 0.46664672
## [13] 0.67019946 0.92749661 0.84314959 0.78829928 0.76964856 0.29398580
## [19] 0.61877522 0.98924359
```

4. In this exercise, we will see how you can use uniform random numbers to simulate the tossing of a coin - where 0 represents a head, and 1 represents a tail.

   Write a second function called `WHcointoss` which will generate n random 0's and $1's$ with parameter p, based on uniform numbers generated by `WHunif` function, using the seeds 1, 2 and 3. That is, 1's are generated if the corresponding uniform number is less than p, and 0's are generated otherwise.

   Input to the `WHcointoss` function should be `n` and `p`, and the output should be the vector of n coin toss outcomes.

```
WHcointoss <- function(n, p) {
     B <- 1*(WHunif(n, 1, 2, 3)<p)
     B
}
```

5. Obtain 20 coin tosses with parameter $p = .5$ using your `WHcointoss function.`

```
 WHcointoss(20, .5)

##  [1] 1 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0
```

6. Write a function `f` that, with input `x`, returns the value of $f(x) = \log(x) + x$ where $\log(x)$ is the natural logarithm function. Use the `secant` function created in the demonstration to find the solution in the interval $[0.5, 1]$ to the equation $f(x) = 0$.

```
f <- function(x) {
    log(x) + x
}
secant(0.5, 1, f)

## [1] 0.5671433
```

7. Write a function called **myrandom** which takes **n x0** as input and that returns n values from the following random number generator.

   For $j = 1, 2, \ldots, n$,

$$
\begin{aligned}
x_j &= 171\ x_{j-1} \bmod 30269 \\
u_j &= x_j/30269.
\end{aligned}
$$

   where x_0 is the initial seed.

```
myrandom <- function(n, x0) {
    u <- numeric(n)
    x <- x0
    for (i in 1:n) {
    x <- (171*x)%%30269
    u[i] <- x/30269
    }
u
}
```

Evaluate 10 random numbers using **myrandom** and an initial seed of 25.

```
myrandom(10, 25)

##  [1] 0.14123361 0.15094651 0.81185371 0.82698470 0.41438435 0.85972447
##  [7] 0.01288447 0.20324424 0.75476560 0.06491790
```

If a student guesses on a multiple choice test with 4 possible answers for each question, the student will be correct 25% of the time. Write a function called **myguesses** which takes the number of questions **n** as input and returns simulated outcomes (1, for correct and 0, for incorrect) for each of the **n** guesses. Use the function **myrandom** with initial seed 325, and follow the method of the 4th question in the demonstration.

```
myguesses <- function(n) {
    x <- myrandom(n, 325)
    correct <- 1*(x <= .25)
return(correct)
}
```

Try out the **myguesses** function on a test with 20 questions. How many answers were correct?

```
myguesses(20)

##  [1] 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0 1 0
```

4 guesses were correct. (We would have expected 5, so this simulated student was unlucky.)