```
# Navdeep Gill Assignment 3

# 1
prime.list <- function(n) {
    # Begin Function

    # Initial checks of input.
    if (!is.numeric(n)) {
        stop("Parameter must be a numeric")
    }
    if (n < 0) {
        stop("The input must be positive.")
    }
    if (n <= 1) {
        stop("The input must be greater than one.")
    }
    if (n > 10000) {
        stop("The input cannot be greater than 10,000")
    }

    # We will first make a sequence of 2:n since 1 is not
prime.
    prime <- 2:n

    # Counter
    iter <- 1

    while (prime[iter] <= sqrt(n)) {
        # Start while
        prime <- prime[prime%%prime[iter] != 0 | prime ==
prime[iter]]
        iter <- iter + 1
    }  #Finish while

    # Output vector titled 'prime'.
    prime <<- prime
}  #End Function

# 2
int.factor <- function(n) {
    # Begin Function

    # Initial checks of input
    if (!is.numeric(n)) {
        stop("Parameter must be a numeric")
    }
    if (n < 0) {
        stop("The input must be positive.")
    }
    if (n <= 1) {
        stop("The input must be greater than one.")
    }
    if (n > 1e+06) {
        stop("The input cannot be greater than 1,000,000")
```

```
    }

    # Make an empty vector v, which will be filled up
later.
    v = c()

    # Implement previous function,prime.list, into this
function,int.factor
    y = prime.list(n)

    # Counter
    iter = 1

    # This while loop will go through each prime and check
the modulus,i.e,
    # whether it is ==0, which indicates it is divisible by
our number of
    # interest,n. Then, it will append that divisor into
the vector x() until
    # the loop is finished, i.e., we get to the last value
in our vector.
    while (n != 1) {
        # Start while
        if (n%%y[iter] == 0) {
            n = n/y[iter]
            v = append(v, y[iter])
        } else {
            iter = iter + 1
        }
    }  #Finish while

    # Now, we will get the prime factors and their
respective powers.
    m = 0
    prime = c(v[1])
    power = c(1)

    for (j in 2:length(v)) {
        # Start for loop
        if (v[j] == v[j - 1]) {
            m = length(power)
            power[m] = power[m] + 1
        } else {
            prime = append(prime, v[j])
            power = append(power, 1)
        }
    }  #Finish for loop

    # Use rbind to put factors and powers together into a
matrix.
    y = rbind(prime, power)
    print(y)

}  #End Function
```

```
# 3
inside.triangle <- function(t, p) {
    # Begin Function

    # Initial checks of input
    if (!is.vector(p)) {
        stop("parameter must be a vector")
    }
    if (!is.numeric(p)) {
        stop("parameter must be numeric")
    }
    if (!is.matrix(t)) {
        stop("parameter must be a vector")
    }
    if (!is.numeric(t)) {
        stop("parameter must be numeric")
    }
    if (length(p) < 2) {
        stop("vector is not long enough")
    }
    if (length(p) > 2) {
        stop("vector is too long")
    }
    if (dim(t) != c(3, 2)) {
        stop("matrix is not the right dimensions")
    }

    # Set up points for triangle based on matrix, t.
    x = c(t[1, 1], t[1, 2])
    y = c(t[2, 1], t[2, 2])
    z = c(t[3, 1], t[3, 2])

    # Move origin to one vertex
    yx = y - x
    zx = z - x
    px = p - x

    # Calculate scalar used for weights wx,wy, and wz.
    d = yx[1] * zx[2] - zx[1] * yx[2]
    print(d)

    # Compute Barycentric weights
    wx = (px[1] * (yx[2] - zx[2]) + px[2] * (zx[1] - yx[1])
+ (yx[1] * zx[2]) -
        (zx[1] * yx[2]))/d
    wy = (px[1] * zx[2] - px[2] * zx[1])/d
    wz = (px[2] * yx[1] - px[1] * yx[2])/d

    print(wx)
    print(wy)
    print(wz)

    # If all weights are between 0 and 1, then p is in
triangle.
    if ((wx > 0) && (wx < 1) && (wy > 0) && (wy < 1) && (wz
```

```
> 0) && (wz < 1)) {
        return(TRUE)
    } else {
        return(FALSE)
    }
}  #End Function
```