```
# Navdeep Gill Assignment 1

# 1
pos.ind <- function(x) {
    if (!is.vector(x)) {
        stop("parameter must be a vector")
    }
    if (!is.numeric(x)) {
        stop("parameter must be numeric")
    }

    n = length(x)
    where.pos = numeric(length(x))
    for (i in 1:n) {
        if (x[i] > 0) {
            where.pos[i] = i
        }
    }
    where.pos = (where.pos[where.pos > 0])
    print(where.pos)
}

# Test Case
x <- c(34, -5, 0, 12, -7, 4)
pos.ind(x)
```

```
## [1] 1 4 6
```

```
# 2
variance <- function(x) {
    if (!is.vector(x)) {
        stop("parameter must be a vector")
    }
    if (!is.numeric(x)) {
        stop("parameter must be numeric")
    }

    sum = 0
    n = length(x)
    for (i in 1:n) {
        sum = sum + x[i]
        x.bar = sum/n
    }
    numerator = numeric(n)
    for (i in 1:n) {
        numerator[i] = (x[i] - x.bar)^2
    }
    var = sum(numerator)/(n - 1)
    print(var)
}

# Test Case
x <- c(1:20)
variance(x)
```

```
## [1] 35
```

```
var(x)
```

```
## [1] 35
```

```
# 3
x <- c(1, 4, 7)
rep(x, rep(3, 3))
```

```
## [1] 1 1 1 4 4 4 7 7 7
```

```
# 4
order <- function(n) {
    if (length(n) > 1) {
        stop("parameter must be vector of length one")
    }
    if (!is.numeric(n)) {
        stop("parameter must be numeric")
    }
    if (n < 0) {
        print("Error. Need a positive number")
    } else {
        n = n
        m = n - (n - 1)
        first = seq(m, n, by = 1)
        second = seq(n, m, by = -1)
        second <- second[-1]
        y <<- c(first, second)
        return(y)
    }
}

# Test Case
order(5)
```

```
## [1] 1 2 3 4 5 4 3 2 1
```

```
# 5
sum.order <- function(y) {
    if (!is.numeric(y)) {
        stop("parameter must be numeric")
    }
    k = length(y)
    sum = 0
    for (i in 1:k) {
        sum = sum + y[i]
    }
    return(sum)
}

# Test Case
sum.order(y)
```

```
## [1] 25
```

```
# The sum is the square of the number, n. So, for n=5 the
sum is 25.

# 6
hist.bin <- function(x, bounds) {
    if (!is.vector(x)) {
        stop("parameter must be a vector")
    }
    if (!is.numeric(x)) {
        stop("parameter must be numeric")
    }
    if (!is.vector(bounds)) {
        stop("bound parameter must be a vector")
    }
    if (!is.numeric(bounds)) {
        stop("bound parameter must be numeric")
    }

    n = diff(bounds)

    if (!(all(n >= 0)) || !(all(!n < 0))) {
        print("Error! Bounds is not strictly monotonic.")
    } else if (all(n >= 0)) {
        # Bin vector x by vector bounds
        bounds1 <<- append(-Inf, bounds)
        bounds2 <<- append(bounds1, Inf)
        bin = cut(x, bounds2, right = FALSE)
        bin.table <<- table(bin)
        bin.count <<- unname(bin.table)
    } else if (all(n < 0)) {
        # Bin vector x by vector bounds
        bounds1 <<- append(-Inf, bounds)
        bounds2 <<- append(bounds1, Inf)
        bin = cut(x, bounds2, right = FALSE)
        bin.table <<- table(bin)
        bin.count <<- unname(bin.table)
    }
}

# Test Cases
x <- c(-4, 10, 5, 24, 12, 34)
bounds = c(0, 2.5, 10)
hist.bin(x, bounds)
bin.count
```

```
## [1] 1 0 1 4
```

```
# Check when bounds is not striclty monotonic
bound_notMono = c(0, 2.5, 10, 9, 8)
hist.bin(x, bound_notMono)
```

```
## [1] "Error! Bounds is not strictly monotonic."
```