Functions

Rafael Sanjuan

Ejercicio 1.

Crea una función impares que dado un vector devuelva la cantidad de elementos impares que contiene.

```
impares <- function(vector) {
  # Funciona realizando la operacion de modulo de 2 sobre todos los elementos del
  # vector. Esto retorna el valor 1 si el elemento es impar o 0 si es par.
  # Sumando todos los 1 que confirman que un elemento es impar obtenemos el num
  # de elementos impares.
  return(sum(vector %% 2))
}

num_impares <- impares(1:5)

print(paste("Hay", num_impares, "impares en el rango [1,5]."))
```

```
## [1] "Hay 3 impares en el rango [1,5]."
```

Ejercicio 2.

Crea una función cambio que dada una matriz de numeros enteros reemplaze todos los NA por el valor 0.

```
cambio <- function(matriz) {
  # Indexa los elementos que cumplen la condicion is.na en la matriz y los
  # remplaza asignando un 0.
  matriz[is.na(matriz)] <- 0
  cambiada <- matriz
  return(cambiada)
}

m_ejemplo <- matrix(NA, nrow=4, ncol=3)
cambio(m_ejemplo)
```

```
##      [,1] [,2] [,3]
## [1,]    0    0    0
## [2,]    0    0    0
## [3,]    0    0    0
## [4,]    0    0    0
```

Ejercicio 3.

Crea una función unir que dados dos vectores devuelva un nuevo vector con los elementos de ambos vectores sin repetidos.

```
unir <- function(v1, v2) {
  # Usamos la operacion de union de conjunto que es aplicable a vectores.
  return(union(v1, v2))
}
```

```r
unir(1:7, 5:10)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

Ejercicio 4.

Crea una función vyc que dado un string devuelva una lista de dos componentes que contenga las vocales y las consonantes.

```r
vyc <- function(frase) {
  # Separamos en un vector de letras.
  vector_letras <- strsplit(frase, split="")[[1]]

  # Obtenemos las vocales
  vocales <- vector_letras[grep("[aeiou]", vector_letras)]

  # Obtengo las consonantes negando las vocales y algunos caracteres.
  consonantes <- vector_letras[grep("[^aeiou ,.]", vector_letras)]

  return(append(list(vocales), list(consonantes)))
}

vyc("Hello wolrd")
```

```
## [[1]]
## [1] "e" "o" "o"
##
## [[2]]
## [1] "H" "l" "l" "w" "l" "r" "d"
```

Ejercicio 5.

Crea una función partir que dado un vector v y dos valores x e y (siendo y opcional), retorne un vector con los valores que aparecen luego del primer x y hasta el primer y. De no indicarse el valor de y se devolveran todos los valores que aparecen luego del primer x hasta el final del vector

```r
partir <- function(v, x, y=NULL) {
  # Inicializamos un vector vacio al que le encadenaremos el resultado
  res = c()

  # Variables que controlaran si aniadir elementos al resultado o no.
  aniadir_siguientes <- FALSE
  x_encontrado <- FALSE

  # Recorremos todos los elementos del vector
  for (i in 1:length(v)) {
    if(!x_encontrado && v[i] == x) {
      # Si no habiamos encontrado x antes y es el elemento actual, indicamos
      # que los siguientes elementos se aniadiran al resultado
      aniadir_siguientes <- TRUE
      x_encontrado <- TRUE
    } else if (!is.null(y) && v[i] == y) {
      # Si y ha sido indicado y es el elemento actual paramos de incluir
```

```
      # elementos al resultado.
      aniadir_siguientes <- FALSE
    } else if (aniadir_siguientes) {
      # Si los elementos estan entre un x y un y se aniaden al resultado.
      res <- c(res, v[i])
    }
  }
  return(res)
}

partir(1:10, 4, 8)
```

```
## [1] 5 6 7
```