Exercise 2

2.3

First Question part a

```
a <- c(5, 12, 32, 50, 10000)
a
```

```
[1]     5    12    32    50 10000
```

First Question part b

```
b <- 1:100
b
```

```
  [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
 [19]  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
 [37]  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
 [55]  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
 [73]  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
 [91]  91  92  93  94  95  96  97  98  99 100
```

OR

```
b <- seq(1, 100)
b
```

```
  [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
 [19]  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
 [37]  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
 [55]  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
 [73]  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
 [91]  91  92  93  94  95  96  97  98  99 100
```

OR

```
b <- seq(1, 100, by=1)
b
```

```
  [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
 [19]  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
 [37]  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
 [55]  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
 [73]  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
 [91]  91  92  93  94  95  96  97  98  99 100
```

OR

```r
b <- seq(1, 100, length.out = 100)
b
```

```
  [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
 [19]  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
 [37]  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
 [55]  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
 [73]  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
 [91]  91  92  93  94  95  96  97  98  99 100
```

First Question part c

```r
c <- seq(2, 100, by=2)
c
```

```
 [1]   2   4   6   8  10  12  14  16  18  20  22  24  26  28  30  32  34  36  38
[20]  40  42  44  46  48  50  52  54  56  58  60  62  64  66  68  70  72  74  76
[39]  78  80  82  84  86  88  90  92  94  96  98 100
```

First Question part d

```r
d <- rep(c(3, 6, 9), c(10, 20, 30))
d
```

```
 [1] 3 3 3 3 3 3 3 3 3 3 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 9 9 9 9 9 9 9 9
[39] 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9
```

Second Question

```r
e <- seq(1, 10, by=1)
e
```

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

OR

```r
e <- 1:10
e
```

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

OR

```r
e <- seq(1, 10, length.out = 10)
e
```

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

Third Question

```r
f <- rep(1:4, times=3)
f
```

```
 [1] 1 2 3 4 1 2 3 4 1 2 3 4
```

Fourth Question

```r
n <- 1:100
sqrt_n <- sqrt(n)
sqrt_n
```

```
  [1]  1.000000  1.414214  1.732051  2.000000  2.236068  2.449490  2.645751
  [8]  2.828427  3.000000  3.162278  3.316625  3.464102  3.605551  3.741657
 [15]  3.872983  4.000000  4.123106  4.242641  4.358899  4.472136  4.582576
 [22]  4.690416  4.795832  4.898979  5.000000  5.099020  5.196152  5.291503
 [29]  5.385165  5.477226  5.567764  5.656854  5.744563  5.830952  5.916080
 [36]  6.000000  6.082763  6.164414  6.244998  6.324555  6.403124  6.480741
 [43]  6.557439  6.633250  6.708204  6.782330  6.855655  6.928203  7.000000
 [50]  7.071068  7.141428  7.211103  7.280110  7.348469  7.416198  7.483315
 [57]  7.549834  7.615773  7.681146  7.745967  7.810250  7.874008  7.937254
 [64]  8.000000  8.062258  8.124038  8.185353  8.246211  8.306624  8.366600
 [71]  8.426150  8.485281  8.544004  8.602325  8.660254  8.717798  8.774964
 [78]  8.831761  8.888194  8.944272  9.000000  9.055385  9.110434  9.165151
 [85]  9.219544  9.273618  9.327379  9.380832  9.433981  9.486833  9.539392
 [92]  9.591663  9.643651  9.695360  9.746794  9.797959  9.848858  9.899495
 [99]  9.949874 10.000000
```

OR

```r
sqrt_n <- sqrt(1:100)
sqrt_n
```

```
  [1]  1.000000  1.414214  1.732051  2.000000  2.236068  2.449490  2.645751
  [8]  2.828427  3.000000  3.162278  3.316625  3.464102  3.605551  3.741657
 [15]  3.872983  4.000000  4.123106  4.242641  4.358899  4.472136  4.582576
 [22]  4.690416  4.795832  4.898979  5.000000  5.099020  5.196152  5.291503
 [29]  5.385165  5.477226  5.567764  5.656854  5.744563  5.830952  5.916080
 [36]  6.000000  6.082763  6.164414  6.244998  6.324555  6.403124  6.480741
 [43]  6.557439  6.633250  6.708204  6.782330  6.855655  6.928203  7.000000
 [50]  7.071068  7.141428  7.211103  7.280110  7.348469  7.416198  7.483315
```

```
 [57]  7.549834  7.615773  7.681146  7.745967  7.810250  7.874008  7.937254
 [64]  8.000000  8.062258  8.124038  8.185353  8.246211  8.306624  8.366600
 [71]  8.426150  8.485281  8.544004  8.602325  8.660254  8.717798  8.774964
 [78]  8.831761  8.888194  8.944272  9.000000  9.055385  9.110434  9.165151
 [85]  9.219544  9.273618  9.327379  9.380832  9.433981  9.486833  9.539392
 [92]  9.591663  9.643651  9.695360  9.746794  9.797959  9.848858  9.899495
 [99]  9.949874 10.000000
```

OR

```
sqrt_n <- sqrt(seq(1, 100))
sqrt_n
```

```
  [1]  1.000000  1.414214  1.732051  2.000000  2.236068  2.449490  2.645751
  [8]  2.828427  3.000000  3.162278  3.316625  3.464102  3.605551  3.741657
 [15]  3.872983  4.000000  4.123106  4.242641  4.358899  4.472136  4.582576
 [22]  4.690416  4.795832  4.898979  5.000000  5.099020  5.196152  5.291503
 [29]  5.385165  5.477226  5.567764  5.656854  5.744563  5.830952  5.916080
 [36]  6.000000  6.082763  6.164414  6.244998  6.324555  6.403124  6.480741
 [43]  6.557439  6.633250  6.708204  6.782330  6.855655  6.928203  7.000000
 [50]  7.071068  7.141428  7.211103  7.280110  7.348469  7.416198  7.483315
 [57]  7.549834  7.615773  7.681146  7.745967  7.810250  7.874008  7.937254
 [64]  8.000000  8.062258  8.124038  8.185353  8.246211  8.306624  8.366600
 [71]  8.426150  8.485281  8.544004  8.602325  8.660254  8.717798  8.774964
 [78]  8.831761  8.888194  8.944272  9.000000  9.055385  9.110434  9.165151
 [85]  9.219544  9.273618  9.327379  9.380832  9.433981  9.486833  9.539392
 [92]  9.591663  9.643651  9.695360  9.746794  9.797959  9.848858  9.899495
 [99]  9.949874 10.000000
```

Fifth Question

```
g <- seq(1, 100, by = 0.4)
g
```

```
  [1]  1.0  1.4  1.8  2.2  2.6  3.0  3.4  3.8  4.2  4.6  5.0  5.4  5.8  6.2  6.6
 [16]  7.0  7.4  7.8  8.2  8.6  9.0  9.4  9.8 10.2 10.6 11.0 11.4 11.8 12.2 12.6
 [31] 13.0 13.4 13.8 14.2 14.6 15.0 15.4 15.8 16.2 16.6 17.0 17.4 17.8 18.2 18.6
 [46] 19.0 19.4 19.8 20.2 20.6 21.0 21.4 21.8 22.2 22.6 23.0 23.4 23.8 24.2 24.6
 [61] 25.0 25.4 25.8 26.2 26.6 27.0 27.4 27.8 28.2 28.6 29.0 29.4 29.8 30.2 30.6
 [76] 31.0 31.4 31.8 32.2 32.6 33.0 33.4 33.8 34.2 34.6 35.0 35.4 35.8 36.2 36.6
 [91] 37.0 37.4 37.8 38.2 38.6 39.0 39.4 39.8 40.2 40.6 41.0 41.4 41.8 42.2 42.6
[106] 43.0 43.4 43.8 44.2 44.6 45.0 45.4 45.8 46.2 46.6 47.0 47.4 47.8 48.2 48.6
[121] 49.0 49.4 49.8 50.2 50.6 51.0 51.4 51.8 52.2 52.6 53.0 53.4 53.8 54.2 54.6
[136] 55.0 55.4 55.8 56.2 56.6 57.0 57.4 57.8 58.2 58.6 59.0 59.4 59.8 60.2 60.6
[151] 61.0 61.4 61.8 62.2 62.6 63.0 63.4 63.8 64.2 64.6 65.0 65.4 65.8 66.2 66.6
[166] 67.0 67.4 67.8 68.2 68.6 69.0 69.4 69.8 70.2 70.6 71.0 71.4 71.8 72.2 72.6
[181] 73.0 73.4 73.8 74.2 74.6 75.0 75.4 75.8 76.2 76.6 77.0 77.4 77.8 78.2 78.6
[196] 79.0 79.4 79.8 80.2 80.6 81.0 81.4 81.8 82.2 82.6 83.0 83.4 83.8 84.2 84.6
[211] 85.0 85.4 85.8 86.2 86.6 87.0 87.4 87.8 88.2 88.6 89.0 89.4 89.8 90.2 90.6
[226] 91.0 91.4 91.8 92.2 92.6 93.0 93.4 93.8 94.2 94.6 95.0 95.4 95.8 96.2 96.6
[241] 97.0 97.4 97.8 98.2 98.6 99.0 99.4 99.8
```

2.6 Filtering vectors based on conditions

11)

```
x <- c(80, 39, NA, 51, 51, 11, NA, NA, NA, 100, 80, 70)
```

a).

```
nonMissings <- x[!is.na(x)] # since is.na command gives us whether the value is a missing value or not
nonMissings
```

```
[1]  80  39  51  51  11 100  80  70
```

b).

```
missingsOdd <- x[(x %% 2) != 0] # since missing values are already not even numbers
missingsOdd
```

```
[1] 39 NA 51 51 11 NA NA NA
```

c).

```
odd <- x[!is.na(x) & (x %% 2) != 0] # extracting only the values which are not odd
#ignoring missing values
odd
```

```
[1] 39 51 51 11
```

d).

```
x <- c(80, 39, NA, 51, 51, 11, NA, NA, NA, 100, 80, 70)
notIn <- x[!is.na(x) &!(x %in% 1:50)] # extracting only the values which not in the set 1:50
notIn
```

```
[1]  80  51  51 100  80  70
```

2.7 Modify a vector

12)

```
age <- c(20, 30, 40, 41, 32, 32, 25,  NA, NA, -4, -6, 9999, 10000)
```

a).

```
a <- replace(age, which(age < 0), NA) # assigning NA to negative values
a
```

```
 [1]    20    30    40    41    32    32    25    NA    NA    NA    NA  9999
[13] 10000
```

b).

```
age <- c(20, 30, 40, 41, 32, 32, 25,  NA, NA, -4, -6, 9999, 10000)
age [age < 0] <- 0 # assigning zero for all negative values

valid <- age[age %in% 1:100] # extracting the valid responses

mean(valid, rm.na=TRUE)
```

```
[1] 31.42857
```

13)

```
set.seed(17620212)
b <- rnorm(100)
b
```

```
  [1]  0.589528488 -0.662937204  0.238279278  0.183757174 -0.002364399
  [6]  0.289002107  0.258796402  0.982174159  0.378628085  0.015035037
 [11] -1.203312799  1.510436562  0.219368378 -0.642429444 -0.373969124
 [16] -0.239829685 -0.186344734  0.517975563 -1.256355393  1.067433297
 [21]  1.035128935 -1.016002843  0.830122365  0.427420672  0.170429825
 [26] -0.001345883 -1.022893025 -0.908602635  0.502535054  0.315929086
 [31]  1.294309571 -0.303323749 -0.322819573 -1.377566743  2.714915313
 [36] -0.512573266  1.342424819 -0.457104082 -1.593015886 -0.202338403
 [41]  1.079678527  0.456102666  1.504041202  0.378318229 -0.289765109
 [46]  1.019989890  0.665591385 -1.076213455  0.272375584  0.545493842
 [51]  0.052391342 -0.402364688  0.152662598 -1.486745812  0.102018231
 [56] -0.024357072  0.068276667  0.075642814  0.379600455 -0.988308679
 [61]  0.701330674 -0.491165150  1.494498791 -1.773934043 -0.460454009
 [66]  0.752256616  0.039189614 -0.939203562 -0.419716046  0.084067026
```

```
 [71] -1.081303093  0.780827145  0.207575277  0.733234796 -0.660969465
 [76]  1.649316796  0.491550464 -0.864054075 -0.919522275 -0.727913488
 [81]  1.197400462 -1.645388340  1.704924934 -1.650667045  0.377823148
 [86]  1.436377659 -1.143144414  0.789086285  1.049357974  2.163786809
 [91]  1.626306920  1.317779758  1.647449733  0.588881226 -0.177613835
 [96]  0.081191404  0.093051240  1.202918954 -1.783424334  0.725816313
```

a).

```
b[1:5] <- 1 # changing the first five values to 1
b
```

```
  [1]  1.000000000  1.000000000  1.000000000  1.000000000  1.000000000
  [6]  0.289002107  0.258796402  0.982174159  0.378628085  0.015035037
 [11] -1.203312799  1.510436562  0.219368378 -0.642429444 -0.373969124
 [16] -0.239829685 -0.186344734  0.517975563 -1.256355393  1.067433297
 [21]  1.035128935 -1.016002843  0.830122365  0.427420672  0.170429825
 [26] -0.001345883 -1.022893025 -0.908602635  0.502535054  0.315929086
 [31]  1.294309571 -0.303323749 -0.322819573 -1.377566743  2.714915313
 [36] -0.512573266  1.342424819 -0.457104082 -1.593015886 -0.202338403
 [41]  1.079678527  0.456102666  1.504041202  0.378318229 -0.289765109
 [46]  1.019989890  0.665591385 -1.076213455  0.272375584  0.545493842
 [51]  0.052391342 -0.402364688  0.152662598 -1.486745812  0.102018231
 [56] -0.024357072  0.068276667  0.075642814  0.379600455 -0.988308679
 [61]  0.701330674 -0.491165150  1.494498791 -1.773934043 -0.460454009
 [66]  0.752256616  0.039189614 -0.939203562 -0.419716046  0.084067026
 [71] -1.081303093  0.780827145  0.207575277  0.733234796 -0.660969465
 [76]  1.649316796  0.491550464 -0.864054075 -0.919522275 -0.727913488
 [81]  1.197400462 -1.645388340  1.704924934 -1.650667045  0.377823148
 [86]  1.436377659 -1.143144414  0.789086285  1.049357974  2.163786809
 [91]  1.626306920  1.317779758  1.647449733  0.588881226 -0.177613835
 [96]  0.081191404  0.093051240  1.202918954 -1.783424334  0.725816313
```

b).

```
length(b) # length of the vector b
```

```
[1] 100
```

```
b[96:100] <- 0 # changing last five values to 0
b
```

```
  [1]  1.000000000  1.000000000  1.000000000  1.000000000  1.000000000
  [6]  0.289002107  0.258796402  0.982174159  0.378628085  0.015035037
 [11] -1.203312799  1.510436562  0.219368378 -0.642429444 -0.373969124
 [16] -0.239829685 -0.186344734  0.517975563 -1.256355393  1.067433297
 [21]  1.035128935 -1.016002843  0.830122365  0.427420672  0.170429825
 [26] -0.001345883 -1.022893025 -0.908602635  0.502535054  0.315929086
 [31]  1.294309571 -0.303323749 -0.322819573 -1.377566743  2.714915313
```

```
 [36] -0.512573266  1.342424819 -0.457104082 -1.593015886 -0.202338403
 [41]  1.079678527  0.456102666  1.504041202  0.378318229 -0.289765109
 [46]  1.019989890  0.665591385 -1.076213455  0.272375584  0.545493842
 [51]  0.052391342 -0.402364688  0.152662598 -1.486745812  0.102018231
 [56] -0.024357072  0.068276667  0.075642814  0.379600455 -0.988308679
 [61]  0.701330674 -0.491165150  1.494498791 -1.773934043 -0.460454009
 [66]  0.752256616  0.039189614 -0.939203562 -0.419716046  0.084067026
 [71] -1.081303093  0.780827145  0.207575277  0.733234796 -0.660969465
 [76]  1.649316796  0.491550464 -0.864054075 -0.919522275 -0.727913488
 [81]  1.197400462 -1.645388340  1.704924934 -1.650667045  0.377823148
 [86]  1.436377659 -1.143144414  0.789086285  1.049357974  2.163786809
 [91]  1.626306920  1.317779758  1.647449733  0.588881226 -0.177613835
 [96]  0.000000000  0.000000000  0.000000000  0.000000000  0.000000000
```

c).

```
b[b > 0.5] <- 1 # assigning 1 to values grater than 0.5
b
```

```
  [1]  1.000000000  1.000000000  1.000000000  1.000000000  1.000000000
  [6]  0.289002107  0.258796402  1.000000000  0.378628085  0.015035037
 [11] -1.203312799  1.000000000  0.219368378 -0.642429444 -0.373969124
 [16] -0.239829685 -0.186344734  1.000000000 -1.256355393  1.000000000
 [21]  1.000000000 -1.016002843  1.000000000  0.427420672  0.170429825
 [26] -0.001345883 -1.022893025 -0.908602635  1.000000000  0.315929086
 [31]  1.000000000 -0.303323749 -0.322819573 -1.377566743  1.000000000
 [36] -0.512573266  1.000000000 -0.457104082 -1.593015886 -0.202338403
 [41]  1.000000000  0.456102666  1.000000000  0.378318229 -0.289765109
 [46]  1.000000000  1.000000000 -1.076213455  0.272375584  1.000000000
 [51]  0.052391342 -0.402364688  0.152662598 -1.486745812  0.102018231
 [56] -0.024357072  0.068276667  0.075642814  0.379600455 -0.988308679
 [61]  1.000000000 -0.491165150  1.000000000 -1.773934043 -0.460454009
 [66]  1.000000000  0.039189614 -0.939203562 -0.419716046  0.084067026
 [71] -1.081303093  1.000000000  0.207575277  1.000000000 -0.660969465
 [76]  1.000000000  0.491550464 -0.864054075 -0.919522275 -0.727913488
 [81]  1.000000000 -1.645388340  1.000000000 -1.650667045  0.377823148
 [86]  1.000000000 -1.143144414  1.000000000  1.000000000  1.000000000
 [91]  1.000000000  1.000000000  1.000000000  1.000000000 -0.177613835
 [96]  0.000000000  0.000000000  0.000000000  0.000000000  0.000000000
```

```
b[b < 0.5] <- 0 # assigning 0 to values less than 0.5
b
```

```
  [1] 1 1 1 1 1 0 0 1 0 0 0 1 0 0 0 0 0 1 0 1 1 0 1 0 0 0 0 0 1 0 1 0 0 0 1 0 1
 [38] 0 0 0 1 0 1 0 0 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 1 0 1
 [75] 0 1 0 0 0 0 1 0 1 0 0 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0
```

d).

```r
set.seed(17620212)
b <- rnorm(100)
b[b > 0.5] <- 1
b[b <= 0.5] <- 0
b
```

```
##   [1] 1 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 1 0 1 1 0 1 0 0 0 0 0 1 0 1 0 0 0 1 0 1
##  [38] 0 0 0 1 0 1 0 0 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 1 0 1
##  [75] 0 1 0 0 0 0 1 0 1 0 0 1 0 1 1 1 1 1 1 1 0 0 0 1 0 1
```

```r
b[b == 0] <- "MALE"
b[b == 1] <- "FEMALE"
b
```

```
##   [1] "FEMALE" "MALE"   "MALE"   "MALE"   "MALE"   "MALE"   "MALE"   "FEMALE"
##   [9] "MALE"   "MALE"   "MALE"   "FEMALE" "MALE"   "MALE"   "MALE"   "MALE"
##  [17] "MALE"   "FEMALE" "MALE"   "FEMALE" "FEMALE" "MALE"   "FEMALE" "MALE"
##  [25] "MALE"   "MALE"   "MALE"   "MALE"   "FEMALE" "MALE"   "FEMALE" "MALE"
##  [33] "MALE"   "MALE"   "FEMALE" "MALE"   "FEMALE" "MALE"   "MALE"   "MALE"
##  [41] "FEMALE" "MALE"   "FEMALE" "MALE"   "MALE"   "FEMALE" "FEMALE" "MALE"
##  [49] "MALE"   "FEMALE" "MALE"   "MALE"   "MALE"   "MALE"   "MALE"   "MALE"
##  [57] "MALE"   "MALE"   "MALE"   "MALE"   "FEMALE" "MALE"   "FEMALE" "MALE"
##  [65] "MALE"   "FEMALE" "MALE"   "MALE"   "MALE"   "MALE"   "MALE"   "FEMALE"
##  [73] "MALE"   "FEMALE" "MALE"   "FEMALE" "MALE"   "MALE"   "MALE"   "MALE"
##  [81] "FEMALE" "MALE"   "FEMALE" "MALE"   "MALE"   "FEMALE" "MALE"   "FEMALE"
##  [89] "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE" "MALE"   "MALE"
##  [97] "MALE"   "FEMALE" "MALE"   "FEMALE"
```

```r
b <- ifelse(b == 0, "MALE", "FEMALE")
b
```

```
  [1] "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE"
  [9] "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE"
 [17] "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE"
 [25] "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE"
 [33] "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE"
 [41] "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE"
 [49] "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE"
 [57] "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE"
 [65] "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE"
 [73] "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE"
 [81] "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE"
 [89] "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE"
 [97] "FEMALE" "FEMALE" "FEMALE" "FEMALE"
```

Or else this can be done by using the following method as well.

```r
# b <- replace(b, which(b==0), "MALE")
# b <- replace(b, which(b==1), " FEMALE")
```

2.8

i) Enter confirmed cases in table 1 to a vector.

```
confirmed_cases <- c(29631, 1151, 1104, 1073, 879, 830, 771, 492, 468, 459, 405, 337, 331,
                     295, 261, 218, 213, 210, 141, 136, 119, 109, 107, 91, 85, 80, 58, 49,
                     49, 36, 18, 18, 10, 1) # Initialize the vector

confirmed_cases # Print the vector
```

```
 [1] 29631  1151  1104  1073   879   830   771   492   468   459   405   337
[13]   331   295   261   218   213   210   141   136   119   109   107    91
[25]    85    80    58    49    49    36    18    18    10     1
```

```
length(confirmed_cases) # To check the length of the vector
```

```
[1] 34
```

```
is.vector(confirmed_cases) # To check if it is a vector
```

```
[1] TRUE
```

```
is.integer(confirmed_cases) # To check if it is a vector of integers
```

```
[1] FALSE
```

```
class(confirmed_cases)
```

```
[1] "numeric"
```

```
typeof(confirmed_cases)
```

```
[1] "double"
```

```
# To check whether all confirmed cases are recorded by checking Total given in Table 1
sum(confirmed_cases)
```

```
[1] 40235
```

ii) Name the elements by province/regions/cities in China.

Modifying the names of an existing vector

```r
# Initialize the vector

names(confirmed_cases) <- c("Hubei", "Guangdong", "Zhejiang", "Henan", "Hunan", "Anhui",
                            "Jiangxi", "Jiangsu", "Chongqing", "Shandong", "Sichuan",
                            "Beijing", "Heilongjiang", "Shanghai", "Fujian", "Hebei",
                            "Shaanxi", "Guangxi", "Yunnan", "Hainan", "Shanxi",
                            "Guizhou", "Liaoning", "Tianjin", "Gansu", "Jilin",
                            "Inner Mongolia", "Ningxia", "Xinjiang", "Hong Kong SAR",
                            "Qinghai", "Taipei and environs", "Macao SAR", "Xizang")


confirmed_cases # Print the vector
```

```
            Hubei         Guangdong          Zhejiang             Henan
            29631              1151              1104              1073
            Hunan             Anhui           Jiangxi           Jiangsu
              879               830               771               492
        Chongqing          Shandong          Sichuan           Beijing
              468               459               405               337
      Heilongjiang          Shanghai           Fujian             Hebei
              331               295               261               218
         Shaanxi          Guangxi           Yunnan            Hainan
              213               210               141               136
          Shanxi          Guizhou          Liaoning           Tianjin
              119               109               107                91
           Gansu            Jilin    Inner Mongolia           Ningxia
              85                80                58                49
        Xinjiang    Hong Kong SAR          Qinghai Taipei and environs
              49                36                18                18
      Macao SAR           Xizang
              10                 1
```

iii) Write Rcodes to answer the following questions.

a) Which province/region/city has the highest number of confirmed cases?

Method 1:

```r
# To get the province/ region/ city which has the highest value from the vector
which.max(confirmed_cases)
```

```
Hubei
    1
```

Method 2:

```r
max(confirmed_cases) # To get the the maximum value/ highest value from the vector
```

```
[1] 29631
```

```
# If equal return as TRUE
# If not return as FALSE
# Check whether each value of the vector and return whether the value is equal to
# the maximum value or not
confirmed_cases == max(confirmed_cases)
```

```
              Hubei          Guangdong           Zhejiang              Henan
              TRUE              FALSE              FALSE              FALSE
             Hunan              Anhui            Jiangxi            Jiangsu
             FALSE              FALSE              FALSE              FALSE
         Chongqing           Shandong            Sichuan            Beijing
             FALSE              FALSE              FALSE              FALSE
      Heilongjiang           Shanghai             Fujian              Hebei
             FALSE              FALSE              FALSE              FALSE
           Shaanxi            Guangxi             Yunnan             Hainan
             FALSE              FALSE              FALSE              FALSE
            Shanxi            Guizhou           Liaoning            Tianjin
             FALSE              FALSE              FALSE              FALSE
             Gansu              Jilin      Inner Mongolia            Ningxia
             FALSE              FALSE              FALSE              FALSE
          Xinjiang      Hong Kong SAR           Qinghai Taipei and environs
             FALSE              FALSE              FALSE              FALSE
         Macao SAR             Xizang
             FALSE              FALSE
```

```
# To select an element which equal to the condition
confirmed_cases[confirmed_cases == max(confirmed_cases)]
```

```
Hubei
29631
```

b) Number of confirmed cases reported in Hebei, China.

```
confirmed_cases['Hebei'] # To select the element with a specific name
```

```
Hebei
  218
```

c) Total number of confirmed cases reported in China.

```
sum(confirmed_cases) # To get the total
```

```
[1] 40235
```

d) Number of cases reported in the capital of China.

```
confirmed_cases['Beijing'] # To select the element with a specific name
```

```
Beijing
    337
```

e) Number of cases reported in Inner Mongolia.

```
confirmed_cases['Inner Mongolia'] # To select the element with a specific name
```

```
Inner Mongolia
            58
```