Name: Mithun G

USN:19BTRCR006

LAB PROGRAM 7

1. In each of the following, determine the final value of answer. Check your result by running the code in R.
a) answer <- 0 for (j in 1:5) answer <- answer + j
b) answer <- NULL for (j in 1:5) answer <- c(answer, j)
c) answer <- 0 for (j in 1:5) answer <- c(answer, j)
d) answer <- 1 for (j in 1:5) answer <- answer * j)

In [3]:

```
# a
answer <- 0
for(j in 1:5){
    answer<- (answer+j)
}
answer
```

15

In [2]:

```
# b
answer <- NULL
for(j in 1:5){
    answer<- c(answer,j)
}
answer
```

1 · 2 · 3 · 4 · 5

In [4]:

```
# c
answer <- 0
for(j in 1:5){
    answer<- c(answer,j)
}
answer
```

0 · 1 · 2 · 3 · 4 · 5

In [5]:

```
# d
answer <- 1
for(j in 1:5){
    answer<- c(answer*j)
}
answer
```

120

2. Does the Eratosthenes() function work properly if n is not an integer? Is an error message required in this case?

In [17]:

```
Eratosthenes <- function(n) {
if (n >= 2) {
     x = seq(2, n)
     prime_nums = c()
     for (i in seq(2, n)) {
         if (any(x == i)) {
             prime_nums = c(prime_nums, i)
             x = c(x[(x %% i) != 0], i)
         }
     }
     return(prime_nums)
 }
 else
 {
     stop("Input number should be at least 2.")
 }
 }
Eratosthenes(12)
Eratosthenes(10.5)
```

2 · 3 · 5 · 7 · 11

2 · 3 · 5 · 7

Answer: The function Eratosthenes() can take input even if it's not an integer, it just considers the whole number part of it

3. Use the idea of the Eratosthenes() function to prove that there are infinitely many primes. Hint: suppose all primes were less than m, and construct a larger value n that would not be eliminated by the sieve.

In [20]:

```
Eratosthenes(1) # we see that there should be minimum 2 to get the sieve
```

```
Error in Eratosthenes(1): Input number should be at least 2.
Traceback:

1. Eratosthenes(1)
2. stop("Input number should be at least 2.")   # at line 15 of file <text>
```

In [21]:

```
Eratosthenes(2000) # we see that this can go infintely many primes, hence proved
```

```
 2 ·  3 ·  5 ·  7 ·  11 ·  13 ·  17 ·  19 ·  23 ·  29 ·  31 ·  37 ·  41 ·  43 ·  47 ·  53 ·
 59 ·  61 ·  67 ·  71 ·  73 ·  79 ·  83 ·  89 ·  97 ·  101 ·  103 ·  107 ·  109 ·  113 ·
 127 ·  131 ·  137 ·  139 ·  149 ·  151 ·  157 ·  163 ·  167 ·  173 ·  179 ·  181 ·  191 ·
 193 ·  197 ·  199 ·  211 ·  223 ·  227 ·  229 ·  233 ·  239 ·  241 ·  251 ·  257 ·  263 ·
 269 ·  271 ·  277 ·  281 ·  283 ·  293 ·  307 ·  311 ·  313 ·  317 ·  331 ·  337 ·  347 ·
 349 ·  353 ·  359 ·  367 ·  373 ·  379 ·  383 ·  389 ·  397 ·  401 ·  409 ·  419 ·  421 ·
 431 ·  433 ·  439 ·  443 ·  449 ·  457 ·  461 ·  463 ·  467 ·  479 ·  487 ·  491 ·  499 ·
 503 ·  509 ·  521 ·  523 ·  541 ·  547 ·  557 ·  563 ·  569 ·  571 ·  577 ·  587 ·  593 ·
 599 ·  601 ·  607 ·  613 ·  617 ·  619 ·  631 ·  641 ·  643 ·  647 ·  653 ·  659 ·  661 ·
 673 ·  677 ·  683 ·  691 ·  701 ·  709 ·  719 ·  727 ·  733 ·  739 ·  743 ·  751 ·  757 ·
 761 ·  769 ·  773 ·  787 ·  797 ·  809 ·  811 ·  821 ·  823 ·  827 ·  829 ·  839 ·  853 ·
 857 ·  859 ·  863 ·  877 ·  881 ·  883 ·  887 ·  907 ·  911 ·  919 ·  929 ·  937 ·  941 ·
 947 ·  953 ·  967 ·  971 ·  977 ·  983 ·  991 ·  997 ·  1009 ·  1013 ·  1019 ·  1021 ·
 1031 ·  1033 ·  1039 ·  1049 ·  1051 ·  1061 ·  1063 ·  1069 ·  1087 ·  1091 ·  1093 ·
 1097 ·  1103 ·  1109 ·  1117 ·  1123 ·  1129 ·  1151 ·  1153 ·  1163 ·  1171 ·  1181 ·
 1187 ·  1193 ·  1201 ·  1213 ·  1217 ·  1223 ·  1229 ·  1231 ·  1237 ·  1249 ·  1259 ·
 1277 ·  1279 ·  1283 ·  1289 ·  1291 ·  1297 ·  1301 ·  1303 ·  1307 ·  1319 ·  1321 ·
 1327 ·  1361 ·  1367 ·  1373 ·  1381 ·  1399 ·  1409 ·  1423 ·  1427 ·  1429 ·  1433 ·
 1439 ·  1447 ·  1451 ·  1453 ·  1459 ·  1471 ·  1481 ·  1483 ·  1487 ·  1489 ·  1493 ·
 1499 ·  1511 ·  1523 ·  1531 ·  1543 ·  1549 ·  1553 ·  1559 ·  1567 ·  1571 ·  1579 ·
 1583 ·  1597 ·  1601 ·  1607 ·  1609 ·  1613 ·  1619 ·  1621 ·  1627 ·  1637 ·  1657 ·
 1663 ·  1667 ·  1669 ·  1693 ·  1697 ·  1699 ·  1709 ·  1721 ·  1723 ·  1733 ·  1741 ·
 1747 ·  1753 ·  1759 ·  1777 ·  1783 ·  1787 ·  1789 ·  1801 ·  1811 ·  1823 ·  1831 ·
 1847 ·  1861 ·  1867 ·  1871 ·  1873 ·  1877 ·  1879 ·  1889 ·  1901 ·  1907 ·  1913 ·
 1931 ·  1933 ·  1949 ·  1951 ·  1973 ·  1979 ·  1987 ·  1993 ·  1997 ·  1999
```

4. Write an R function called compound.interest() which computes this amount. Your function should have three arguments.

In [24]:

```
compound.interest <- function(price, rate, n)
{
    total_interest <- 0
    for (i in n){
          total_interest <- total_interest + (price*(1+(rate)^n)/100)
}
    cat("The compound interest calculated is:",total_interest)
}

compound.interest(2000,3,5)
```

```
The compound interest calculated is: 4880
```

5. Consider the inbuilt data set “cars”.
a) Find Correlation between possible variables and pairwise correlation.
b) Find regression line between appropriate variables.
c) Display the summary statistics and comment on the results.

In [26]:

```
df <- datasets::cars
head(df)
```

A data.frame: 6 × 2

	speed	dist
	<dbl>	<dbl>
1	4	2
2	4	10
3	7	4
4	7	22
5	8	16
6	9	10

In [28]:

```
# finding correaltion between possible variables and pairwise correaltion
cor(df$speed, df$dist)
```

0.80689490068921

In [29]:

```
# Find regression line between appropriate variables.
model <- lm(dist ~ speed, data=df)    # this gives the best fit line's parameters
print(model)
```

```
Call:
lm(formula = dist ~ speed, data = df)

Coefficients:
(Intercept)        speed
    -17.579        3.932
```

In [30]:

```
summary(model)
```

```
Call:
lm(formula = dist ~ speed, data = df)

Residuals:
    Min      1Q  Median      3Q     Max
-29.069  -9.525  -2.272   9.215  43.201

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -17.5791     6.7584  -2.601   0.0123 *
speed         3.9324     0.4155   9.464 1.49e-12 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 15.38 on 48 degrees of freedom
Multiple R-squared:  0.6511,    Adjusted R-squared:  0.6438
F-statistic: 89.57 on 1 and 48 DF,  p-value: 1.49e-12
```

From the summary() we can see that:

- the Linear model we used has a R-squared error of 0.6438 *(R squared error ranges from 0 to 1, higher the value better is the fit of the line)*
- The linear model has F-statistic of 89.57 on 1