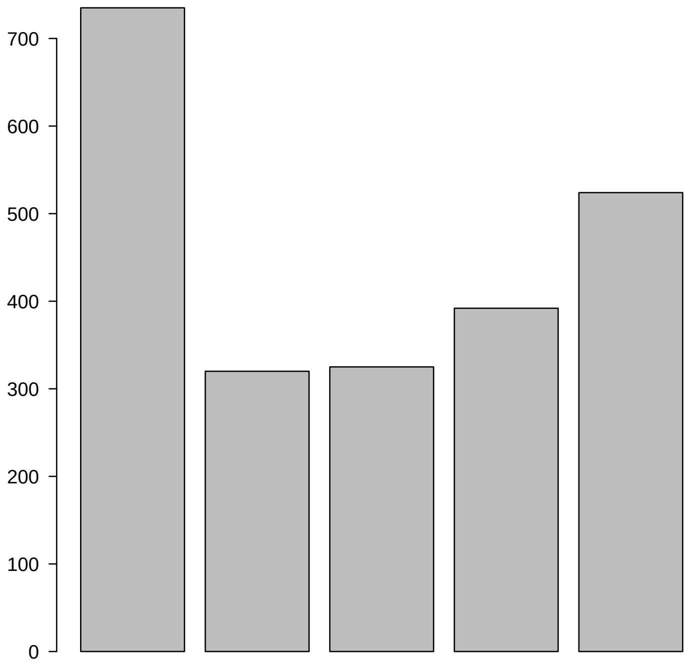
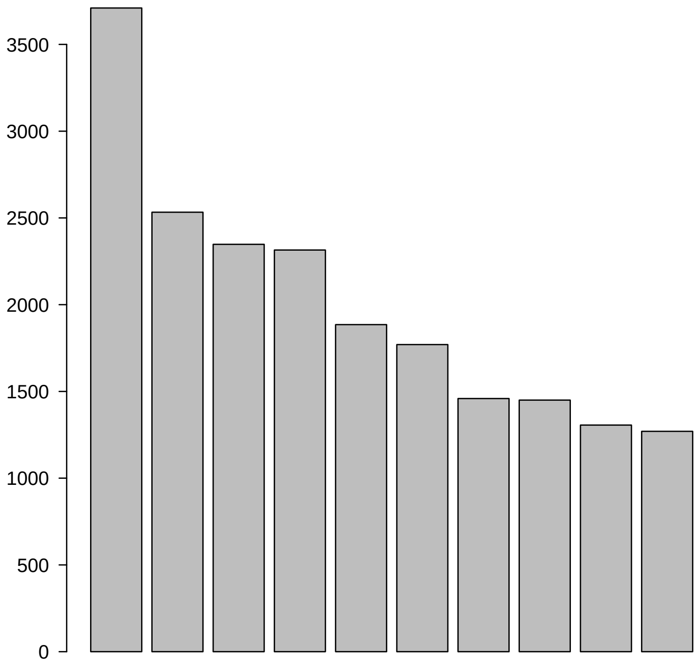
```r
##Question 1

#a.
11 * 11

## [1] 121

#Answer is 121

#b.
11 * 111

## [1] 1221

#Answer is 1221

#c.
11 * 1111

## [1] 12221

#Answer is 12221

#d.
11 * 11111

## [1] 122221

#Answer is 122221

#e.
#Answer: Based on the pattern I see above, I can safely predict that the product of
# 11 and 1111111111111111111 is 12222222222222222221

#f.
options(digits=15)
11 * 1111111111111111111

## [1] 12222222222222223360

#Answer: 12222222222222223360, I am the one who is right. The number shown is because R cannot
#handle such numbers.

##Question 2

#a.
riversYards <- rivers * 1760

#b.
riversYards[1:10]

##  [1] 1293600  563200  572000  689920  922240  792000 2567840  237600  818400 1056000
```

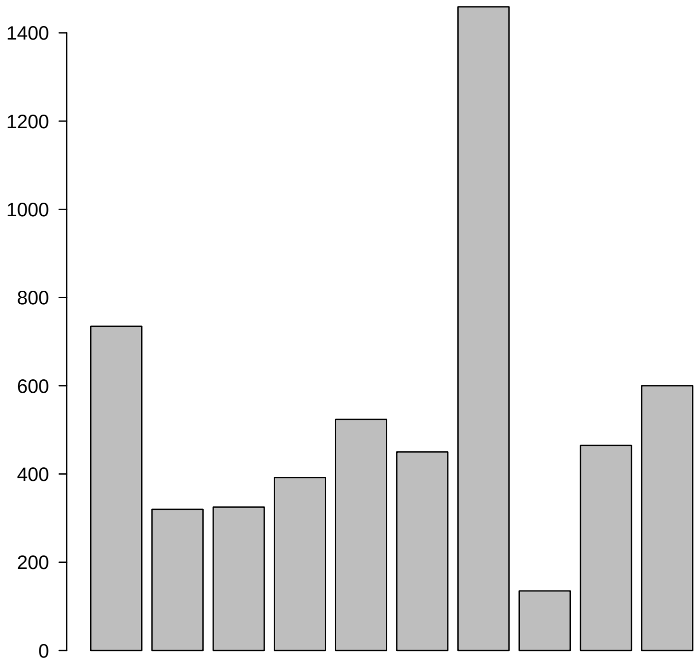
```
#Answer: 1293600  563200  572000  689920  922240  792000 2567840  237600  818400 1056000

#c.
riversBetween <- riversYards[(riversYards<1000000) & (riversYards>500000)]
riversBetween

##  [1] 563200 572000 689920 922240 792000 818400 580800 591360 554400 579040 510400 888800
## [13] 616000 716320 503360 924000 686400 575520 633600 538560 686400 739200 512160 598400
## [25] 619520 827200 616000 528000 985600 584320 721600 809600 758560 616000 594880 880000
## [37] 723360 765600 862400 545600 809600 674080 660000 959200 783200 668800 528000 668800
## [49] 663520 748000 739200 616000 633600 946880 552640 633600 950400 746240 545600 528000
## [61] 781440 529760 924000 633600 931040 880000 756800

#d.
barplot(rivers[1:5])
```



```
#Answer: No, it is not recorded in a decreasing order.

#e.
Rivers <- sort(rivers, decreasing = TRUE)
barplot(Rivers[1:10])
```



```
#Answer: There are 4 rivers that are longer than 2000 miles.

#f.
barplot(rivers[1:10])
```



```
rivers[1:10]

##  [1]  735  320  325  392  524  450 1459  135  465  600

#Answer: There are 0 river that are longer than 2000 miles.

#g.
#Answer: There are 0 river that are longer than 1500 miles.

#h.
#Answer: There are 3 rivers that are longer than 500 miles.

##Question 3

#3a. Generate this code pattern:
## [1] 1 4 9 16 25 36 49 64 81 100
#Answer is:
(1:10)^2

##  [1]   1   4   9  16  25  36  49  64  81 100

#3b. Generate this code pattern:
## [1] 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32
## [17] 34 36 38 40 42 44 46 48 50 52 54 56 58 60 62 64
## [33] 66 68 70 72 74 76 78 80 82 84 86 88 90 92 94 96
## [49] 98 100
#Answer is:
seq(2, 100, 2)

##  [1]   2   4   6   8  10  12  14  16  18  20  22  24  26  28  30  32  34  36  38  40  42
## [22]  44  46  48  50  52  54  56  58  60  62  64  66  68  70  72  74  76  78  80  82  84
## [43]  86  88  90  92  94  96  98 100

#3c.
#Answer:
n <- 1:5
an <- 1 - (1/n)
an #[1] 0.0000000 0.5000000 0.6666667 0.7500000 0.8000000

## [1] 0.000000000000000 0.500000000000000 0.666666666666667 0.750000000000000
## [5] 0.800000000000000

#3d.
#Answer:
n <- 5
an <- 1 - (1/n)
an #[1] 0.8

## [1] 0.8

# it seems that R gave me a much less sig fig number for this one.


#3e.
#Answer:
#i:
sum(1:200) #[1] 20100
```

```
## [1] 20100

#ii:
nVec = 1:20
GLseq <- (-1)^(nVec+1)/(2*nVec-1)
sum(GLseq) #[1] 0.772906

## [1] 0.77290595166696

pi/4 #[1] 0.7853982

## [1] 0.785398163397448

nVec2 = 1:100
GLseq <- (-1)^(nVec2+1)/(2*nVec2-1)
sum(GLseq) #[1] 0.7828982

## [1] 0.782898225889638

pi/4 #[1] 0.7853982

## [1] 0.785398163397448

nVec2 = 1:1000
GLseq <- (-1)^(nVec2+1)/(2*nVec2-1)
sum(GLseq) #[1] 0.7851482

## [1] 0.785148163459948

pi/4 #[1] 0.7853982

## [1] 0.785398163397448

nVec2 = 1:10000
GLseq <- (-1)^(nVec2+1)/(2*nVec2-1)
sum(GLseq) #[1] 0.7853732

## [1] 0.785373163397511

pi/4 #[1] 0.7853982

## [1] 0.785398163397448

nVec2 = 1:100000
GLseq <- (-1)^(nVec2+1)/(2*nVec2-1)
sum(GLseq) #[1] 0.7853957

## [1] 0.785395663397448

pi/4 #[1] 0.7853982

## [1] 0.785398163397448

nVec2 = 1:1000000
GLseq <- (-1)^(nVec2+1)/(2*nVec2-1)
sum(GLseq) #[1] 0.7853979

## [1] 0.785397913397448
```

```
pi/4 #[1] 0.7853982
```

```
## [1] 0.785398163397448
```

```
#Answer: as n is from 1:higher & higher number, we can expect the Gregory formula to simplify
#closer to pi/4.

#3f.
#Answer
#i:
vec = rep(c(5:1,1:5),10)
rep(seq(1,100), vec)
```

```
##   [1]   1   1   1   1   1   2   2   2   2   3   3   3   4   4   5   6   7   7   8   8   8
##  [22]   9   9   9   9  10  10  10  10  10  11  11  11  11  11  12  12  12  12  13  13  13
##  [43]  14  14  15  16  17  17  18  18  18  19  19  19  19  20  20  20  20  20  21  21  21
##  [64]  21  21  22  22  22  22  23  23  23  24  24  25  26  27  27  28  28  28  29  29  29
##  [85]  29  30  30  30  30  30  31  31  31  31  31  32  32  32  32  33  33  33  34  34  35
## [106]  36  37  37  38  38  38  39  39  39  39  40  40  40  40  40  41  41  41  41  41  42
## [127]  42  42  42  43  43  43  44  44  45  46  47  47  48  48  48  49  49  49  49  50  50
## [148]  50  50  50  51  51  51  51  51  52  52  52  52  53  53  53  54  54  55  56  57  57
## [169]  58  58  58  59  59  59  59  60  60  60  60  60  61  61  61  61  61  62  62  62  62
## [190]  63  63  63  64  64  65  66  67  67  68  68  68  69  69  69  69  70  70  70  70  70
## [211]  71  71  71  71  71  72  72  72  72  73  73  73  74  74  75  76  77  77  78  78  78
## [232]  79  79  79  79  80  80  80  80  80  81  81  81  81  81  82  82  82  82  83  83  83
## [253]  84  84  85  86  87  87  88  88  88  89  89  89  89  90  90  90  90  90  91  91  91
## [274]  91  91  92  92  92  92  93  93  93  94  94  95  96  97  97  98  98  98  99  99  99
## [295]  99 100 100 100 100 100
```

```
#ii:
2^(seq(0,8))
```

```
## [1]   1   2   4   8  16  32  64 128 256
```

```
#iii:
rep(rep(seq(3,7),c(3,2,4,2,1)),5)
```

```
##  [1] 3 3 3 4 4 5 5 5 5 6 6 7 3 3 3 4 4 5 5 5 5 6 6 7 3 3 3 4 4 5 5 5 5 6 6 7 3 3 3 4 4 5 5
## [44] 5 5 6 6 7 3 3 3 4 4 5 5 5 5 6 6 7
```

```
#iv:
seq(-6,21,by=3)
```

```
##  [1] -6 -3  0  3  6  9 12 15 18 21
```

```
sessionInfo()
```

```
## R version 4.2.2 (2022-10-31)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Ventura 13.1
##
## Matrix products: default
## LAPACK: /Library/Frameworks/R.framework/Versions/4.2/Resources/lib/libRlapack.dylib
##
```

```
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## loaded via a namespace (and not attached):
##  [1] compiler_4.2.2  magrittr_2.0.3  tools_4.2.2     glue_1.6.2      vctrs_0.5.1
##  [6] stringi_1.7.8   highr_0.10      knitr_1.41      stringr_1.5.0   xfun_0.36
## [11] lifecycle_1.0.3 rlang_1.0.6     evaluate_0.19
```

```r
Sys.time()
```

```
## [1] "2023-02-01 09:29:26 PST"
```