The results below are generated from an R script.

```
# Question 1

# Decimal to binary
# a. 0.625
# Answer
# using double dabble method
# Step 1 multiply by 2
# 0.625*2 = 1.25
# Step 2 if number is larger than 1, put 1 if not put 0.
# 0.1--
# use the fractional part of 1.25 and repeat step 1.
# 0.25*2 = 0.5
# 0.10-
# repeat step 1 again to fractional value of 0.5.
# 0.5*2 = 1
# 0.101
# therefore, the decimal value of 0.625 to binary is 0.101 of base 2.

# b. 1/9 (0.11111...)
# we only take some decimal place since it is not efficient to show all of the numbers
# 0.11111*2 = 0.22222 = 0.0-------
# 0.22222*2 = 0.44444 = 0.00------
# 0.44444*2 = 0.88888 = 0.000-----
# 0.88888*2 = 1.77776 = 0.0001----
# 0.77776*2 = 1.55552 = 0.00011---
# 0.55552*2 = 1.11104 = 0.000111--
# 0.11104*2 = 0.22208 = 0.0001110-
# 0.22208*2 = 0.44416 = 0.00011100

# we can also keep going since it is repeating, there will be a pattern
# therefore, the decimal value of 1/9 is 0.00011100 (repeated) of base 2.

# binary to decimal
# using positional notation binary
# the 1 and 0 corresponds if the number has a value or not.
# if it is 0, the number will be multiplied by 0.

# a. 1111
1*2^3 + 1*2^2 + 1*2^1 + 1*2^0

## [1] 15

# b. 1010
1*2^3 + 0*2^2 + 1*2^1 + 0*2^0

## [1] 10

# c. 1010.0101
# write out the integer part first.
intPart = 1*2^3 + 0*2^2 + 1*2^1 + 0*2^0
intPart

## [1] 10
```

```r
# now write the fractional part (0.0101)
fracPart = 0*2^(-1) + 1*2^(-2) + 0*2^(-3) + 1*2^(-4)
fracPart
```

```
## [1] 0.3125
```

```r
# the answer will be the int + frac which is
intPart + fracPart
```

```
## [1] 10.312
```

```r
# d. 1.010101...(repeated)
# do the same as c.
intPart = 1*2^1
intPart
```

```
## [1] 2
```

```r
# fractional part (0.010101)
fracPart = 0*2^(-1) + 1*2^(-2) + 0*2^(-3) + 1*2^(-4) + 0*2^(-5) + 1*2^(-6)
fracPart
```

```
## [1] 0.32812
```

```r
# add them all to get answer
intPart + fracPart
```

```
## [1] 2.3281
```

```r
# Question 2
# a. 6/7 (0.8571429)
# 0.8571*2 = 1.7142
# 0.7142*2 = 1.4284
# 0.4284*2 = 0.8568
# 0.8568*2 = 1.7136
# answer: 0.1101

# b. 1/7 (0.1428571)
# 0.1428*2 = 0.2856
# 0.2856*2 = 0.5712
# 0.5712*2 = 1.1424
# 0.1424*2 = 0.2848
# answer: 0.0010

# c.
# 0.1101 + 0.0010 = 0.1111
1*2^(-1) + 1*2^(-2) + 1*2^(-3) + 1*2^(-4)
```

```
## [1] 0.9375
```

```r
# d.
# 110 (6) +  1 (1) = 111 (7)
# 111/111 = 1


# Question 3
```

```r
x = 1000000
y = 999999
A = function(x,y){
  x^4 - y^4
}
B = function(x,y){
  (x^2+y^2)*(x+y)*(x-y)
}
options(digits=15)
A(x,y)
```

```
## [1] 3999993999971581952
```

```r
B(x,y)
```

```
## [1] 3.999994000004e+18
```

```r
# B is more accurate since R cannot handle big number, and the number x^4 and y^4
# is very very big.

# Question 4
# a.
# pretend these numbers are prime numbers (1 to 10000 are primes)
primes = 1:10000
# this code will print last 100 elements of this vector
primes[(10000-100):10000]
```

```
##   [1]  9900  9901  9902  9903  9904  9905  9906  9907  9908  9909  9910  9911  9912  9913
##  [15]  9914  9915  9916  9917  9918  9919  9920  9921  9922  9923  9924  9925  9926  9927
##  [29]  9928  9929  9930  9931  9932  9933  9934  9935  9936  9937  9938  9939  9940  9941
##  [43]  9942  9943  9944  9945  9946  9947  9948  9949  9950  9951  9952  9953  9954  9955
##  [57]  9956  9957  9958  9959  9960  9961  9962  9963  9964  9965  9966  9967  9968  9969
##  [71]  9970  9971  9972  9973  9974  9975  9976  9977  9978  9979  9980  9981  9982  9983
##  [85]  9984  9985  9986  9987  9988  9989  9990  9991  9992  9993  9994  9995  9996  9997
##  [99]  9998  9999 10000
```

```r
#b.
#this code will add up first 9000 numbers in vector primes.
sum(primes[1:9000])
```

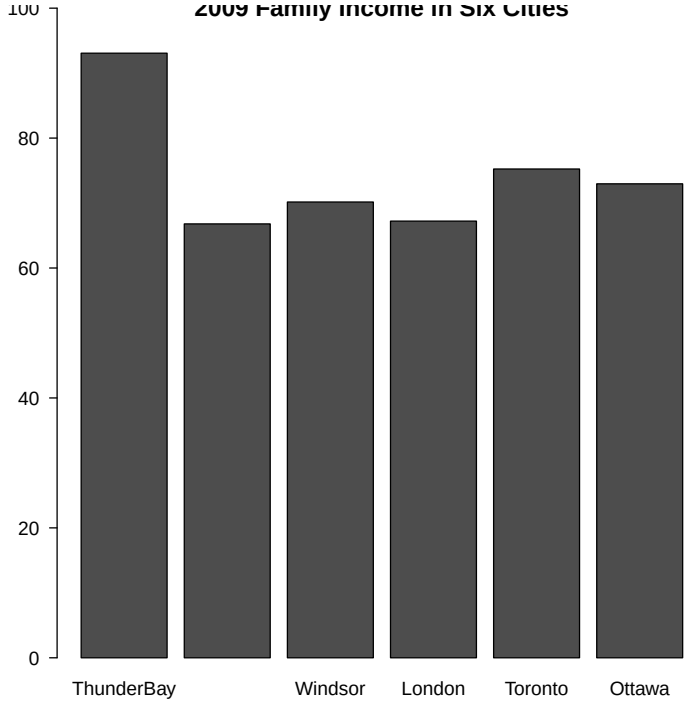
```
## [1] 40504500
```

```r
# Question 5.
x = 100
y = 99
myfunc = function(x,y){
  (x^16)*(((x^8 - y^8) / ((196059601)*(19801)*(199))) - 1)

}
myfunc(x,y)
```

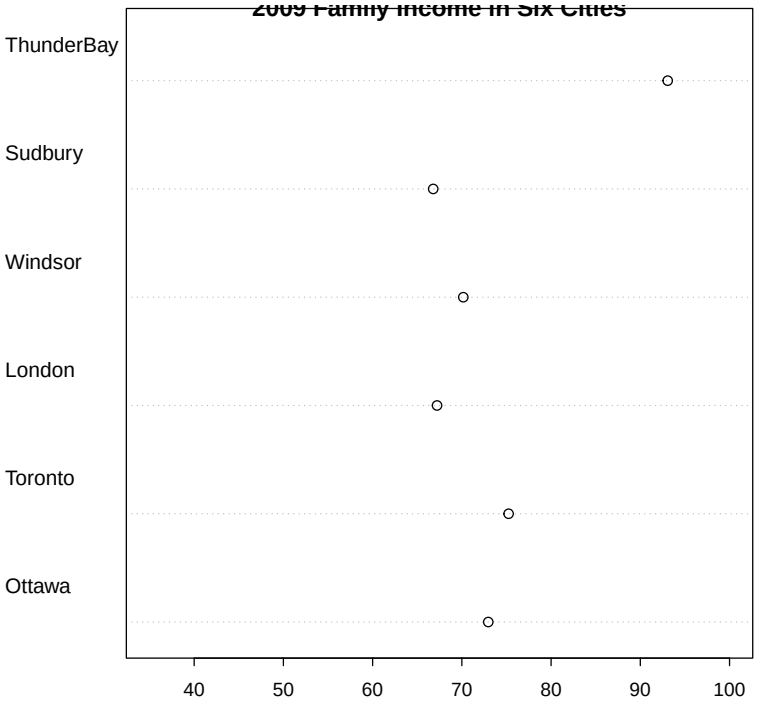
```
## [1] 133226762955018784
```

```r
# Question 6
# to approach this we can use the as.Date library.
tom = as.Date("1999-07-05")
david = as.Date("2003-12-12")

difftime(david, tom, units = "days")
```

```
## Time difference of 1621 days

# 7.
cities = c("ThunderBay", "Sudbury", "Windsor", "London", "Toronto", "Ottawa")
inc = c(93.07, 66.79, 70.16, 67.22, 75.24, 72.96)

income = matrix(inc, nrow = 1, ncol = length(cities))
colnames(income) <- cities

income

##      ThunderBay Sudbury Windsor London Toronto Ottawa
## [1,]      93.07   66.79   70.16  67.22   75.24  72.96

barplot(income, ylim = c(0,100), main = "2009 Family Income in Six Cities")
```




```
dotchart(income, labels = "", xlim = c(35,100), main = "2009 Family Income in Six Cities")
```



The R session information (including the OS info, R version and all packages used):

```
sessionInfo()

## R version 4.2.2 (2022-10-31 ucrt)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 19044)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United States.utf8  LC_CTYPE=English_United States.utf8
## [3] LC_MONETARY=English_United States.utf8 LC_NUMERIC=C
## [5] LC_TIME=English_United States.utf8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## loaded via a namespace (and not attached):
## [1] compiler_4.2.2 tools_4.2.2    tinytex_0.44   highr_0.10     knitr_1.42
## [6] xfun_0.37      evaluate_0.20

Sys.time()

## [1] "2023-02-25 19:56:43 PST"
```