```
import numpy as np
import cvxpy as cp

# Exchange rate data.
tickers = ["USD", "EUR", "GBP", "CAD", "JPY", "CNY", "RUB", "MXN", "INR", "BRL"]
n = len(tickers)
F = np.zeros((n, n))
# USD
data = ([1.0, 0.87, 0.76, 1.31, 108.90, 6.72, 65.45, 19.11, 71.13, 3.69],
# EUR
[1.0, 0.88, 1.51, 125.15, 7.72, 75.23, 21.96, 81.85, 4.24],
# GBP
[1.0, 1.72, 142.94, 8.82, 85.90, 25.08, 93.50, 4.84],
# CAD
[1.0, 82.93, 5.11, 49.82, 14.54, 54.23, 2.81],
# JPY
[1.0, 0.062, 0.60, 0.18, 0.65, 0.034],
# CNY
[1.0, 9.74, 2.85, 10.61, 0.55],
# RUB
[1.0, 0.29, 1.09, 0.056],
# MXN
[1.0, 3.73, 0.19],
# INR
[1.0, 0.052],
# BRL
[1.0])
for i in range(n):
    F[i,i:] = data[i]
for j in range(n):
    for i in range(j+1,n):
        F[i,j] = 1.035/F[j,i]

# Initial and final portfolios.
c_req = np.arange(1,n+1)
c_req = 1e4*c_req/c_req.sum()
c_init = c_req[::-1]

print("c_init: ", c_init)
print("c_req: ", c_req)
print("Exchange rate matrix :", F)
```

```
c_init:  [1818.18181818 1636.36363636 1454.54545455 1272.72727273 1090.90909091
  909.09090909  727.27272727  545.45454545  363.63636364  181.81818182]
c_req:  [ 181.81818182  363.63636364  545.45454545  727.27272727  909.09090909
 1090.90909091 1272.72727273 1454.54545455 1636.36363636 1818.18181818]
Exchange rate matrix : [[1.00000000e+00 8.70000000e-01 7.60000000e-01 1.31000000e+00
  1.08900000e+02 6.72000000e+00 6.54500000e+01 1.91100000e+01
  7.11300000e+01 3.69000000e+00]
 [1.18965517e+00 1.00000000e+00 8.80000000e-01 1.51000000e+00
  1.25150000e+02 7.72000000e+00 7.52300000e+01 2.19600000e+01
  8.18500000e+01 4.24000000e+00]
 [1.36184211e+00 1.17613636e+00 1.00000000e+00 1.72000000e+00
  1.42940000e+02 8.82000000e+00 8.59000000e+01 2.50800000e+01
  9.35000000e+01 4.84000000e+00]
 [7.90076336e-01 6.85430464e-01 6.01744186e-01 1.00000000e+00
  8.29300000e+01 5.11000000e+00 4.98200000e+01 1.45400000e+01
  5.42300000e+01 2.81000000e+00]
 [9.50413223e-03 8.27007591e-03 7.24080034e-03 1.24804052e-02
  1.00000000e+00 6.20000000e-02 6.00000000e-01 1.80000000e-01
  6.50000000e-01 3.40000000e-02]
 [1.54017857e-01 1.34067358e-01 1.17346939e-01 2.02544031e-01
  1.66935484e+01 1.00000000e+00 9.74000000e+00 2.85000000e+00
  1.06100000e+01 5.50000000e-01]
 [1.58135982e-02 1.37578094e-02 1.20488941e-02 2.07747892e-02
  1.72500000e+00 1.06262834e-01 1.00000000e+00 2.90000000e-01
  1.09000000e+00 5.60000000e-02]
 [5.41601256e-02 4.71311475e-02 4.12679426e-02 7.11829436e-02
  5.75000000e+00 3.63157895e-01 3.56896552e+00 1.00000000e+00
  3.73000000e+00 1.90000000e-01]
 [1.45508224e-02 1.26450825e-02 1.10695187e-02 1.90853771e-02
  1.59230769e+00 9.75494816e-02 9.49541284e-01 2.77479893e-01
  1.00000000e+00 5.20000000e-02]
 [2.80487805e-01 2.44103774e-01 2.13842975e-01 3.68327402e-01
  3.04411765e+01 1.88181818e+00 1.84821429e+01 5.44736842e+00
  1.99038462e+01 1.00000000e+00]]
```

```
#initializing currency exchange matrix
X = cp.Variable((n,n))

#currency holdings after exchange
ones = np.ones(n)
purchased = cp.matmul((X/F), ones)
sold = cp.matmul(X.T, ones)
c_final = purchased - sold + c_init

#cost function
def total_cost(c):
  currency_values = np.sqrt(F[:,0]/F[0,:])
  currency_decrease = c_init - c
  return currency_values@currency_decrease

#convex problem formulation
prob = cp.Problem(cp.Minimize(total_cost(c_final)),[X >= 0, cp.diag(X) == 0, cp.matmul(X.T
result = prob.solve()

print("The minimum cost is: ", total_cost(c_final.value), " USD")
print("Final Currency Holdings: ", c_final.value)
```

```
The minimum cost is:  7.720059340057868  USD
Final Currency Holdings:  [1257.49577907 1636.36363639 1771.6701903   727.27272732  9
 1090.90909094 1272.72727321 1454.54545464 1636.3636367  1818.1818182 ]
```