LAB 8.1

Scelgo i numeri 1001, 1034 e 1011 come esempi di numeri non primi intorno a 1000. Per calcolare l'ordine dei gruppi Z_{1010*}, Z_{1034*} e Z_{1012*}, applichiamo la funzione phi di Eulero, ottenendo i seguenti risultati:

$$\phi\ (1001)= 1001\times(1-17)\times(1-111)\times(1-\underline{113})=720$$

$$\phi\ (1034)=1034*(1-12)\times(1-111)\times(1-147)=460$$

$$\phi\ (1012)= 1012\times(1-12)\times(1-111)\times(1-123)=440$$

Successivamente, verrà presentato il codice utilizzato e i risultati dei test del programma generato da tale codice, come richiesto.

```python
import random
import math
import time

def check_basic_cases(n):
    """Se numero minore di 2 o divisibile per 2 returniamo true"""
    if n < 2 or n % 2 == 0:
        return False
    return True

def calculate_s_and_q(n):
    s = 0 #step 1
    q = n-1 #step 2
    while q % 2 == 0: #step 3
        s += 1
        q //= 2
    return s, q

def modular_exponentiation(a, q, n):
  #step 5
  x = 1
  qp = 0
  while qp < q:
    qp += 1
    x = (a*x) % n
  return x

def test_primality(n, a):
```

```
  #questa funzione ritorna TRUE se il numero n passato come argomento è
"probabilmente" primo, FALSE se non è primo

  random.seed(time.time())
  # Preliminarmente, vengono gestiti i casi in cui il valore di 'n' sia inferiore a
2 o pari, poiché, in tali circostanze,
  # risulta evidente che 'n' non può essere un numero primo.
  if not check_basic_cases(n):
    return False

  # Qui inizia il vero algoritmo Miller-Rabin step 1 e step 2 step 3
  s, q = calculate_s_and_q(n)

  # step4 , viene campionato casualmente un numero nell'intervallo {2, 3, ..., n-
2}.
  # Definiamo il limite superiore del range come (n-2) e il limite inferiore come
2.
  # Il numero campionato, indicato come 'a', viene selezionato direttamente
all'interno del range specificato
  # senza utilizzare un approccio casuale, poiché si intende usare diversi valori
di 'a' all'interno dell'intervallo.
  # Il valore di 'a' viene quindi fornito come argomento diretto alla funzione.


  # step 5 , viene eseguito il calcolo di a^q % n e il risultato viene assegnato
alla variabile x.
  # L'approccio diretto, calcolando prima a^q e poi applicando il modulo n,
potrebbe risultare inefficiente in determinati scenari.
  x = modular_exponentiation(a, q, n)

  if x == 1 or x == n-1: #step 6
    return True

  while s-1 >= 0: #step 7
    x = pow(x, 2) % n
    if x == n-1: #controllo se x è congruo a -1 mod n
      return True
    s -= 1

  return False #step 8

def MCPrimalityTest(n, a):
    if test_primality(n, a):
        #print(str(n) + " is probably a prime number")
        return True
    else:
        #print(str(n) + " a compound number")
        return False
```

```python
if __name__ == "__main__":
  #numeri scelti sono 1001, 1034 e 1012
  n = 1001
  non_lying_witnes = 0
  print("\nProva con n=1001:")
  for i in range(2, n-1):
    if(not MCPrimalityTest(n, i)):
      non_lying_witnes += 1
  percentage_non_lying_witnes = (non_lying_witnes / (n - 3)) * 100
  print(f"{non_lying_witnes} volte su {n-3} risulta un testimone non bugiardo ({percentage_non_lying_witnes:.2f}%)\n")

  n = 1034
  non_lying_witnes = 0
  print("Prova con n=1034:")
  for i in range(2, n-1):
    if(not MCPrimalityTest(n, i)):
      non_lying_witnes += 1
  percentage_non_lying_witnes = (non_lying_witnes / (n - 3)) * 100
  print(f"{non_lying_witnes} volte su {n-3} risulta un testimone non bugiardo ({percentage_non_lying_witnes:.2f}%)\n")

  n = 1012
  non_lying_witnes = 0
  print("Prova con n=1012:")
  for i in range(2, n-1):
    if(not MCPrimalityTest(n, i)):
      non_lying_witnes += 1
  percentage_non_lying_witnes = (non_lying_witnes / (n - 3)) * 100
  print(f"{non_lying_witnes} volte su {n-3} risulta un testimone non bugiardo ({percentage_non_lying_witnes:.2f}%)")
```

```
Prova con n=1001:
990 volte su 998 risulta un testimone non bugiardo (99.20%)

Prova con n=1034:
1031 volte su 1031 risulta un testimone non bugiardo (100.00%)

Prova con n=1012:
1009 volte su 1009 risulta un testimone non bugiardo (100.00%)
```

Emerge che la numerosità dei testimoni risulta costantemente superiore a quella dei bugiardi. Questo è evidenziato dal fatto che, considerando la percentuale di bugiardi pari a 8/998 ≃ 0,8% in un caso specifico e, di conseguenza, approssimativamente ~99,2% di testimoni, nonché l'assenza totale di bugiardi in altri due scenari, è stato possibile constatare che la frazione di testimoni risulta ampiamente superiore al 50%.