COMP8760

Class Worksheet 2

Sanjay Bhattacherjee

This worksheet contains programming tasks. Your programs should provide an input-output interface in the precise format as detailed for the respective questions under **Sample I/O**.

You may use any programming language of your choice.

1. **Finding Prime Numbers:**

   (a) Write a function that will take an integer n as input and use the *exhaustive trial division method* to claim with certainty if it is a composite or a prime. The function will return

   - 0 if n is composite, and
   - 1 if it is prime.

   Using this function, check if a number that is being input to the program is a prime or not.
   Sample I/O 1:

   ```
   Please enter the integer n:  97
   Searching for divisors between 2 and 9
   The number 97 is a prime.
   ```

   Sample I/O 2:

   ```
   Please enter the integer n:  10201
   Searching for divisors between 2 and 101
   The number 10201 is composite with the certificate of compositeness being 101.
   ```

   (b) Write a function that will take an integer n and the number of iterations $k > 1$ as input and

   - use the *partial trial division* with all $p < 100$, followed by
   - the Fermat's primality testing algorithm with k values of $a > 100$.

   The function will return

   - $a > 1$ as the *"certificate of compositeness"* if n is composite, and
   - 1 if it is prime.

   Using this function,

   i. Generate a 6-digit number p that is probably a prime.
   Sample I/O 1:

   ```
   Checking for n=217290
   Searching for divisors only among primes between 2 and 97
   Checking for n=176283
   Searching for divisors only among primes between 2 and 97
   Checking for n=737230
   Searching for divisors only among primes between 2 and 97
   Checking for n=666942
   Searching for divisors only among primes between 2 and 97
   Checking for n=715646
   Searching for divisors only among primes between 2 and 97
   Checking for n=353721
   Searching for divisors only among primes between 2 and 97
   Checking for n=711616
   Searching for divisors only among primes between 2 and 97
   Checking for n=454164
   Searching for divisors only among primes between 2 and 97
   Checking for n=694691
   Searching for divisors only among primes between 2 and 97
   ```

```
Running Fermat's test for k=50 iterations
Checking for n=299968
Searching for divisors only among primes between 2 and 97
Checking for n=491973
Searching for divisors only among primes between 2 and 97
Checking for n=755612
Searching for divisors only among primes between 2 and 97
Checking for n=349334
Searching for divisors only among primes between 2 and 97
Checking for n=902953
Searching for divisors only among primes between 2 and 97
Running Fermat's test for k=50 iterations
The number 902953 is probably prime (verified with partial trail division with
prime numbers less than 100 and with Fermat's test with 50 iterations).
```

ii. Take an integer input and report if it is prime or composite. If it is composite, output the certificate of compositeness.

Sample I/O 1:

```
Please enter the integer n:  1009
Please enter the number of iterations k:  50
Searching for divisors only among primes between 2 and 97
Running Fermat's test for k=50 iterations
The number 1009 is probably prime (verified with partial trail division with
prime numbers less than 100 and with Fermat's test with 50 iterations).
```

Sample I/O 2:

```
Please enter the integer n:  10201
Please enter the number of iterations k:  50
Searching for divisors only among primes between 2 and 97
Running Fermat's test for k=50 iterations
The number 10201 is composite with the certificate of compositeness being 2842.
```

Note that if the number being tested for primality is divisible by a prime $p < 100$, in that case, that prime serves as the certificate of compositeness.

`for i=0 to k-1 do`
    Choose $a \in [2, \dots, n-1]$ at random
    $b \leftarrow a^{n-1} \mod n$
    `if` $b \neq 1$`, return (Composite, a)`
`return ("Probably Prime", 1)`

Table 1: Fermat's primality testing algorithm.

2. **Extended Euclidean Algorithm**:
We recollect that the extended Euclidean algorithm takes as input two integers a, b and returns three values d, x and y such that

$$\gcd(a, b) = d = x \cdot a + y \cdot b.$$

The algorithm is described as follows.

(a) Write a program that will take as input two integers a, b and run the extended Euclidean algorithm with them. For each iteration of the algorithm, print the following:

i. The computation and result of q
ii. The computation and result of r
iii. The computation and result of s
iv. The computation and result of t
v. The status of the algorithm in the form $r = s \cdot a + t \cdot b$.

After the last iteration, print the equation $d = x \cdot a + y \cdot b$.

Sample I/O:

Input:	a, b
Output:	d, x, y
Initialisation:	
	$r' \leftarrow a,\ r \leftarrow b$
	$s' \leftarrow 1,\ s \leftarrow 0$
	$t' \leftarrow 0,\ t \leftarrow 1$
Iteration:	`while` $r \neq 0$ `do:`
	$q \leftarrow \lfloor r'/r \rfloor$
	$(r', r) \leftarrow (r, r' - q \cdot r)$
	$(s', s) \leftarrow (s, s' - q \cdot s)$
	$(t', t) \leftarrow (t, t' - q \cdot t)$
Final:	$d \leftarrow r', x \leftarrow s', y \leftarrow t'$
	`return` d, x, y

Table 2: The extended Euclidean algorithm for finding the values of d, x, y such that $\gcd(a, b) = d = x \cdot a + y \cdot b$.

```
Please enter the first integer:  22
Please enter the second integer:  7
----------------------
a = 22, b = 7
Initialising:
r1=22, s1=1, t1=0,           r1 = s1xa + t1xb = 1x22 + 0x7 = 22
r=7, s=0, t=1,           r = sxa + txb = 0x22 + 1x7 = 7
----------------------
q = floor(22/7) = 3
r = 1 = 22 - 3x7
s = 1 = 1 - 3x0
t = -3 = 0 - 3x1
sa + tb =1x22 + -3x7 = 1
----------------------
q = floor(7/1) = 7
r = 0 = 7 - 7x1
s = -7 = 0 - 7x1
t = 22 = 1 - 7x-3
sa + tb =-7x22 + 22x7 = 0
----------------------
The GCD of 22 and 7 is 1
1 = 1x22 + -3x7
```

(b) Write a program that will take as input a positive integer N and:

i. Use the extended Euclidean algorithm to find the values of d, s and t in the equation

$$\gcd(a, N) = d = s \cdot a + t \cdot N$$

for all $a \in \mathbb{Z}_N$.

ii. The set $\mathbb{Z}_N^*$ has elements such that $\gcd(a, N) = 1$. For such an a, we can write

$$\gcd(a, N) = s \cdot a = 1 \pmod{N}.$$

List all $a \in \mathbb{Z}_N^*$ and their respective multiplicative inverses s.

Sample I/O:

```
Please enter the integer N: 12
gcd (12, 1) = 1 = 0x12 + 1x1
(inverse of 1 = 1)
gcd (12, 2) = 2 = 0x12 + 1x2
gcd (12, 3) = 3 = 0x12 + 1x3
```

```
gcd (12, 4) = 4 = 0x12 + 1x4
gcd (12, 5) = 1 = -2x12 + 5x5
(inverse of 5 = 5)
gcd (12, 6) = 6 = 0x12 + 1x6
gcd (12, 7) = 1 = 3x12 + -5x7
(inverse of 7 = 7)
gcd (12, 8) = 4 = 1x12 + -1x8
gcd (12, 9) = 3 = 1x12 + -1x9
gcd (12, 10) = 2 = 1x12 + -1x10
gcd (12, 11) = 1 = 1x12 + -1x11
(inverse of 11 = 11)

(Z/12Z)* = [1, 5, 7, 11]
```