Part II Computational Projects 2020

C3318H

Attach to front of report

Project number:

15.1

15.1

Question 1

A program to test for primality using trial division can be found in page 9. We run the script `q1.m` in page 9 to get all the primes in the given intervals.

```
>> q1

primes[1900:1999] = 1901, 1907, 1913, 1931, 1933, 1949, 1951, 1973, 1979, ...
    1987, 1993, 1997, 1999,
primes[1294268500 : 1294268700] =
```

Primes in $[1900, 1999]$: $1901, 1907, 1913, 1931, 1933, 1949, 1951, 1973, 1979, 1987, 1993, 1997, 1999$
Primes in $[1294268500, 1294268700]$: Empty

Question 2

A program to carry out the Fermat test base a can be found in page 9 named `fermat_test.m`. We run the script `q2.m` in page 10 to apply it on the intervals given in Question 1 .

```
>> q2

 primes[1900,1999] = 1901, 1907, 1913, 1931, 1933, 1949, 1951, 1973, 1979, ...
     1987, 1993, 1997, 1999,
pseudoprime(2) = 1905,
pseudoprime(3) = ,
pseudoprime(4) = 1905,
pseudoprime(5) = ,
pseudoprime(6) = ,
pseudoprime(7) = ,
pseudoprime(8) = 1905,
pseudoprime(9) = ,
pseudoprime(10) = ,
pseudoprime(11) = ,
pseudoprime(12) = ,
pseudoprime(13) = ,
primes[1294268500,1294268700] = ,
pseudoprime(2) = ,
pseudoprime(3) = ,
pseudoprime(4) = ,
pseudoprime(5) = ,
pseudoprime(6) = ,
pseudoprime(7) = ,
pseudoprime(8) = ,
pseudoprime(9) = ,
pseudoprime(10) = ,
```

```
pseudoprime(11) = ,
pseudoprime(12) = ,
pseudoprime(13) = ,
```

We notice that in $[1900, 1999]$, the numbers which pass the Fermat test bases $a = 2, 3 \cdots 13$ are $1901, 1907, 1913, 1931, 1933, 1949, 1951, 1973, 1979, 1987, 1993, 1997, 1999$, which are the same numbers we found in question 1.

The Fermat pseudoprime we find is 1905 which is a pseudoprime in base $2, 4$ and 8.

We don't find any prime or pseudprime in $[1294268500, 1294268700]$ which is consistent with our previous result.

Now for avoiding integer overflow, we write each integer $b < 10^{10}$ in the form $10^5.u + v$ s.t $0 \leq u, v < 10^5$, using division algorithm. Then for any $a < 10^{10}$, we have

$$ab \mod N = ((10^5(au \mod N) \mod N) + (av \mod N)) \mod N$$

Since MATLAB can do multiplications of two integers less than 10^{10} and 10^5, we are fine. A program to execute this can be found in page 10 named `mod_mult.m`.

Now we compute $a^b \mod N$ recursively as follows:

1. $a^1 \mod N = a \mod N$

2. If b is even, $a^b \mod N = (a^{b/2} \mod N)^2 \mod N$ where we can do the squaring using `mod_mult.m`.

3. If b is odd, $a^b \mod N = (a.a^{b-1} \mod N) \mod N$, again doing the multiplication using `mod_mult.m`.

A program to execute this can be found in page 10 named `powermod.m`.

Question 3

The code for a function to determine the primes, base 2 pseudoprimes and absolute pseudoprimes up to an integer M can be found in page 10 `q3.m` . We use the following command to list all the base 2 pseudoprimes and absolute pseudoprimes up to 10^6, the results are shown in table 1 and 2 respectively.

```
>> [prime,pseudo2,abs_pseudo] = q3 ( 10^6 );
>> find(pseudo2)
>> find(abs_pseudo)
```

	F.p		F.p		F.p		F.p		F.p		F.p
1	341	42	30889	83	115921	124	258511	165	464185	206	680627
2	561	43	31417	84	121465	125	264773	166	476971	207	683761
3	645	44	31609	85	123251	126	266305	167	481573	208	688213
4	1105	45	31621	86	126217	127	271951	168	486737	209	710533
5	1387	46	33153	87	129889	128	272251	169	488881	210	711361
6	1729	47	34945	88	129921	129	275887	170	489997	211	721801
7	1905	48	35333	89	130561	130	276013	171	493697	212	722201
8	2047	49	39865	90	137149	131	278545	172	493885	213	722261
9	2465	50	41041	91	149281	132	280601	173	512461	214	729061
10	2701	51	41665	92	150851	133	282133	174	513629	215	738541
11	2821	52	42799	93	154101	134	284581	175	514447	216	741751
12	3277	53	46657	94	157641	135	285541	176	526593	217	742813
13	4033	54	49141	95	158369	136	289941	177	530881	218	743665
14	4369	55	49981	96	162193	137	294271	178	534061	219	745889
15	4371	56	52633	97	162401	138	294409	179	552721	220	748657
16	4681	57	55245	98	164737	139	314821	180	556169	221	757945
17	5461	58	57421	99	172081	140	318361	181	563473	222	769567
18	6601	59	60701	100	176149	141	323713	182	574561	223	769757
19	7957	60	60787	101	181901	142	332949	183	574861	224	786961
20	8321	61	62745	102	188057	143	334153	184	580337	225	800605
21	8481	62	63973	103	188461	144	340561	185	582289	226	818201
22	8911	63	65077	104	194221	145	341497	186	587861	227	825265
23	10261	64	65281	105	196021	146	348161	187	588745	228	831405
24	10585	65	68101	106	196093	147	357761	188	604117	229	838201
25	11305	66	72885	107	204001	148	367081	189	611701	230	838861
26	12801	67	74665	108	206601	149	387731	190	617093	231	841681
27	13741	68	75361	109	208465	150	390937	191	622909	232	847261
28	13747	69	80581	110	212421	151	396271	192	625921	233	852481
29	13981	70	83333	111	215265	152	399001	193	635401	234	852841
30	14491	71	83665	112	215749	153	401401	194	642001	235	873181
31	15709	72	85489	113	219781	154	410041	195	647089	236	875161
32	15841	73	87249	114	220729	155	422659	196	653333	237	877099
33	16705	74	88357	115	223345	156	423793	197	656601	238	898705
34	18705	75	88561	116	226801	157	427233	198	657901	239	915981
35	18721	76	90751	117	228241	158	435671	199	658801	240	916327
36	19951	77	91001	118	233017	159	443719	200	665281	241	934021
37	23001	78	93961	119	241001	160	448921	201	665333	242	950797
38	23377	79	101101	120	249841	161	449065	202	665401	243	976873
39	25761	80	104653	121	252601	162	451905	203	670033	244	983401
40	29341	81	107185	122	253241	163	452051	204	672487	245	997633
41	30121	82	113201	123	256999	164	458989	205	679729		

Table 1: F.p = Fermat pseudoprimes base 2

	a.f.p		a.f.p		a.f.p		a.f.p		a.f.p		a.f.p
1	561	9	15841	17	101101	25	294409	33	512461	41	838201
2	1105	10	29341	18	115921	26	314821	34	530881	42	852841
3	1729	11	41041	19	126217	27	334153	35	552721	43	997633
4	2465	12	46657	20	162401	28	340561	36	656601		
5	2821	13	52633	21	172081	29	399001	37	658801		
6	6601	14	62745	22	188461	30	410041	38	670033		
7	8911	15	63973	23	252601	31	449065	39	748657		
8	10585	16	75361	24	278545	32	488881	40	825265		

Table 2: a.f.p = absolute Fermat pseudoprimes

If an integer N is pseudoprime in base a and b, then we have

$$(ab)^{N-1} \equiv a^{N-1}b^{N-1} \equiv 1 \pmod{N}$$

hence a pseudoprime base ab as well. So it's enough to check only for prime bases.

We define the *highest pseudo base* of a number n to be the largest prime q s.t it's a pseudoprime base all primes up to q. Now for given $M > 0$, the function in page 11 named `highest_pseudo_base.m` returns the highest pseudo base (at most 11) for each $0 < n < M$.

We run the following command

```
>> highest_pseudo_base = highest_pseudo_base ( 10^6 );

>> find(highest_pseudo_base == 7)

        ans =

        721801

>> find(highest_pseudo_base == 11)

        ans =

        1  0  empty double row vector
```

So only 721801 passes the test for bases $2, 3, 5, 7$ and no number for bases $2, 3, 5, 7, 11$. So we have to test for a up to 11.

Question 4

A code to evaluate the Jacobi symbol and carry out the Euler Test can be found in page 12 and 12 named `jacobi_symbol.m` and `euler_test.m` respectively.

Code of a function to determine the primes, base 2 Euler pseudoprimes and absolute Euler pseudoprimes up to an integer M can be found in page 12 named `q4.m`. We use the following command to list all the base 2 Euler pseudoprimes and absolute Euler pseudoprimes up to 10^6.

```
>> [prime,pseudo2,abs_pseudo] = q4 ( 10^6);
```

```
>> find(pseudo2)

>> find(abs_pseudo)

        ans =

        1  0  empty double row vector
```

We see that there is no absolute Euluer pseudoprime in this range. Base 2 Euler pseudoprimes are shown in table 3.

	E.p		E.p		E.p		E.p		E.p		E.p
1	561	20	30121	39	115921	58	266305	77	486737	96	721801
2	1105	21	33153	40	126217	59	271951	78	488881	97	741751
3	1729	22	34945	41	129921	60	278545	79	489997	98	745889
4	1905	23	41041	42	130561	61	280601	80	493697	99	748657
5	2047	24	42799	43	149281	62	294409	81	514447	100	800605
6	2465	25	46657	44	158369	63	314821	82	526593	101	818201
7	3277	26	49141	45	162401	64	323713	83	530881	102	825265
8	4033	27	52633	46	164737	65	334153	84	552721	103	838201
9	4681	28	62745	47	172081	66	340561	85	580337	104	838861
10	6601	29	65281	48	188057	67	348161	86	588745	105	841681
11	8321	30	74665	49	196093	68	357761	87	625921	106	852481
12	8481	31	75361	50	208465	69	390937	88	635401	107	852841
13	10585	32	80581	51	215265	70	399001	89	647089	108	873181
14	12801	33	85489	52	220729	71	410041	90	656601	109	875161
15	15841	34	87249	53	223345	72	427233	91	658801	110	877099
16	16705	35	88357	54	233017	73	448921	92	665281	111	916327
17	18705	36	90751	55	252601	74	449065	93	670033	112	976873
18	25761	37	104653	56	253241	75	458989	94	683761	113	983401
19	29341	38	113201	57	256999	76	476971	95	711361	114	997633

Table 3: E.p = Euler pseudoprime base 2

If an integer N is Euler pseudoprime in base a and b, then it's pseudoprime in base ab since Jacobi Symbol is multiplicative. Hence it's enough to check only for prime bases.

Just like we did for Fermat test, we can define *highest pseudo base* analogously for Euler test. Also we only need to check the numbers which are base 2 pseudoprimes but not absolute pseudoprimes, which we can produce from `q4.m`. For such numbers, the function in page 13 named `q4_2.m` gives the largest prime q (at most 13) such that it's a pseudprime base all primes up to q.

We run the following command

```
>> highest_pseudo_base = q4_2 ( 10^6 );
>> find( highest_pseudo_base == 11 )

        ans =

        399001

>> find( highest_pseudo_base == 13 )

        ans =
```

```
        1  0  empty double row vector
```

Question 5

A code to carry out the Strong Test can be found in page 13 named `strong_test` respectively.

Code of a function to determine the primes, base 2 Strong pseudoprimes and absolute Strong pseudoprimes up to an integer M can be found in page 14 named `q5.m`. We use the following command to list all the base 2 Strong pseudoprimes and absolute Strong pseudoprimes up to 10^6.

```
>> [prime,pseudo2,abs_pseudo] = q5 ( 10^6 );
>> find(pseudo2)
>> find(abs_pseudo)

        ans =

        1  0  empty double row vector
```

We see that there is no absolute Strong pseudoprime less than 10^6. Base 2 Strong pseudoprimes are shown in table 3.

	S.p		S.p		S.p		S.p		S.p		S.p
1	2047	9	49141	17	104653	25	271951	33	489997	41	838861
2	3277	10	52633	18	130561	26	280601	34	514447	42	873181
3	4033	11	65281	19	196093	27	314821	35	580337	43	877099
4	4681	12	74665	20	220729	28	357761	36	635401	44	916327
5	8321	13	80581	21	233017	29	390937	37	647089	45	976873
6	15841	14	85489	22	252601	30	458989	38	741751	46	983401
7	29341	15	88357	23	253241	31	476971	39	800605		
8	42799	16	90751	24	256999	32	486737	40	818201		

Table 4: S.p = Strong pseudoprime base 2

We only need to check the numbers which are base 2 pseudoprimes, which we can produce from `q5.m`. For such numbers, the function in page 15 named `q5_2.m` gives the largest prime q (at most 3) such that it's a pseudprime base q.

We run the following command.

```
>> find(highest_pseudo == 3)

ans =

1  0  empty double row vector
```

So it's enough to check for $a = 2, 3$.

Question 6

We run the script `q6.m` in page 15 to produce Tables 5, 6 and 7 which shows the number of primes and the number of Fermat/Euler/strong pseudoprimes base 2 in the intervals $[10^k, 10^k + 10^5]$

for $5 \leq k \leq 9$ when $a = 2$(Table 5), $a = 3$(Table 6) and both $a = 2, 3$(Table 7) respectively.

k	primes	Fermat 2-pseudoprimes	Euler 2-pseudoprimes	Strong 2-pseudoprimes
5	8392	28	13	3
6	7216	16	9	4
7	6241	6	4	0
8	5411	1	0	0
9	4832	0	0	0

Table 5: #primes and #Fermat/Euler/strong pseudoprimes base 2

k	primes	Fermat 2-pseudoprimes	Euler 2-pseudoprimes	Strong 2-pseudoprimes
5	8392	28	16	7
6	7216	13	9	3
7	6241	5	4	0
8	5411	1	1	1
9	4832	0	0	0

Table 6: #primes and #Fermat/Euler/strong pseudoprimes base 3

k	primes	Fermat 2-pseudoprimes	Euler 2-pseudoprimes	Strong 2-pseudoprimes
5	8392	9	3	0
6	7216	4	2	0
7	6241	2	2	0
8	5411	1	0	0
9	4832	0	0	0

Table 7: #primes and #Fermat/Euler/strong pseudoprimes base both 2 and 3

We see that in each row of every table, numbers are decreasing from left to right. This suggests that Euler test is stronger than Fermat test and Strong test is the strongest among all 3.

Question 7

We know that if a number $1 \leq N \leq 10^5$ passes the Strong test base 2 and 3, then it's a prime. So using that, we can easily find all the primes up to 10^5. Now for numbers $10^5 \leq N \leq 10^10$, we first apply the strong test base 2 and 3. If it passes, we simply apply the trial division, since this is the only way we know so far to ensure that a number is prime.

Now if we are given a large number of integers to check for primality, we can list all the primes up to 10^5 using Strong test base 2 and 3. Now for a given N,

1. If $1 \leq N \leq 10^5$, we can just check whether it exists in list.

2. For $10^5 < N \leq 10^10$, we apply Strong test base 2 and 3. If it passes, we just keep checking whether it's divisible by any prime starting from 2 up to $\sqrt{N}$. Since $\sqrt{N} \leq 10^5$, we do have all the primes up to $\sqrt{N}$ in our list. This is gonna be a lot faster than trial division.

The script `q7.m` in page 17 runs this method and trial division on randomly chosen 10000 integers in the given range and measures the time taken by them. The result is shown below

```
>> q7
best method
Elapsed time is 1.792599 seconds.
trial division
Elapsed time is 3.116340 seconds.
```

We can see that our method too about 57% of the time taken by trial division. The difference will be more apparent if we try on a larger number of integers.

Question 8

We take $k = 17$. The script `q8.m` in page 18 generates t random values of a for each odd $2^{k-1} < N < 2^k$ and counts the number of composite N s.t N passes the strong test for all those values of a. Some of the results are shown in table 8 for $t = 1, 2, \cdots, 5$.

t	x	$x+y$	$100x/(x+y)$
1	10	5719	0.175162
1	7	5716	0.122613
1	6	5715	0.105097
2	0	5709	0.000000
2	0	5709	0.000000
2	0	5709	0.000000
3	0	5709	0.000000
3	0	5709	0.000000
3	0	5709	0.000000
4	0	5709	0.000000
4	0	5709	0.000000
4	0	5709	0.000000
5	0	5709	0.000000
5	0	5709	0.000000
5	0	5709	0.000000

Table 8:
x = number of composites passing the test
y = number of primes in the range
$100x/(x+y) = \mathbb{P}(\ N$ is composite $\mid N$ passes t rounds of the strong test)

Programms

Listing 1: `prime_td.m`

```
function p = prime_td(N)

if N == 1
    p = 0;
elseif N == 2
    p = 1;
else
    i = 2;
    p = 1;
    while i <= sqrt(N)
        if  mod(N,i) == 0
            p = 0 ;
            break
        else
            i=i+1 ;
        end
    end
end
end
```

Listing 2: `q1.m`

```
fprintf('primes[1900:1999] = ')
for i = 1900:1999
    if prime_td(i) == 1
        fprintf('%d, ',i);
    end
end

fprintf('\n primes[1294268500 : 1294268700] = ')
for j = 1294268500 : 1294268700
    if prime_td(j) == 1
        fprintf('%d, ',j)
    end
end
```

Listing 3: `fermat_test.m`

```
function p = fermat_test(a,N)

if a > N-1
    error('Error. \n a= %d must be less than N = %d ',a,N )

elseif gcd(a,N) > 1
    p = 0;

elseif powermod(a,N-1,N) == 1
    p = 1;

else
    p = 0;

end

end
```

Listing 4: `q2.m`

```
for uv = [1900 1294268500 ; 1999 1294268700 ]
    u = uv(1);
    v = uv(2);

    A = zeros(13,v-u+1);
    primes = ones(1,v-u+1);

    for a = 2:13
        for N = u:v
            A(a,N-u+1) = fermat_test(a,N) ;
            primes(N-u+1) = primes(N-u+1) * fermat_test(a,N) ;
        end
    end

    fprintf('\n primes[%d,%d] = ',u,v)
    fprintf('%d, ',find(primes)-1+u )

    for a = 2:13
        fprintf('\n pseudoprime(%d) = ',a)
        fprintf('%d, ', find( A(a,:) - primes )-1+u )
    end

end
```

Listing 5: `mod_mult.m`

```
function m = mod_mult(a,b,N)

if b < 10^5
    m = mod(a*b,N);
else
    u = floor(b/10^5);
    v = b - 10^5*u;
    m = mod( 10^5*(mod(a*u,N)) , N) + mod(a*v,N);
    m = mod(m,N);
end
```

Listing 6: `powermod.m`

```
function r = powermod(a,b,N)

if b == 1
    r = mod(a,N);
else
    r = powermod(a,floor(b/2),N);
    r = mod_mult(r,r,N);

    if mod(b,2) == 1
        r = mod_mult(r,a,N);
    end
end
end
```

Listing 7: `q3.m`

```
function [prime,pseudo2,abs_pseudo] = q3 ( M )

prime = zeros(1,M);
```

```
prime(2) = 1;
pseudo2 = zeros(1,M);
abs_pseudo = zeros(1,M);

for N = 3:M
    if fermat_test(2,N) == 1
        prime(N) = 1 ;

        p = 2;
        while p < sqrt(N)+1
            if prime(p) == 1
                if mod(N,p) == 0
                    prime(N) = 0 ;
                    pseudo2(N) = 1 ;

                    %%%%%%%%%%% Test for absolute pseudprime %%%%%%%%%%%
                    abs_pseudo(N) = 1;
                    q = 3;
                    while q < sqrt(N)
                        if prime(q) == 1
                            if mod(N,q) ~= 0
                                if fermat_test(q,N) == 0
                                    abs_pseudo(N) = 0;
                                    break
                                else
                                    q = q+1 ;
                                end
                            else
                                q = q+1 ;
                            end
                        else
                            q = q+1;
                        end
                    end
                    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

                    break
                else
                    p = p+1;
                end
            else
                p = p+1;
            end
        end
    end
end

end
```

Listing 8: `highest_pseudo_base.m`

```
function highest_pseudo_base = highest_pseudo_base ( M )

[~,pseudo2,abs_pseudo] = q3 ( M );
q = [ 2 3 5 7 11 ];
highest_pseudo_base = 2.*(pseudo2 - abs_pseudo) ;

for N = 1:M
    if highest_pseudo_base(N) == 2
        for i = 2:5
            if fermat_test(q(i),N) == 1
                highest_pseudo_base(N) = q(i) ;
```

```
            else
                break
            end
        end
    end
end
```

Listing 9: `jacobi_symbol.m`

```
function r = jacobi_symbol(M,N)

M = mod(M,N);

if gcd(M,N) > 1
    r = 0;
elseif M == 1
    r = 1;
elseif mod(M,2) == 0
    r = (-1)^(((mod(N,16))^2-1)/8)*jacobi_symbol(M/2,N);
else
    r = (-1)^(mod(M-1,8)*mod(N-1,8)/4)*jacobi_symbol(N,M);
end
```

Listing 10: `euler_test.m`

```
function p = euler_test(a,N)

if mod(N,2) == 0
    p = 0;
elseif gcd(a,N) > 1
    p = 0;
elseif powermod(a,(N-1)/2,N) == mod(jacobi_symbol(a,N),N)
    p = 1;
else
    p = 0;
end
end
```

Listing 11: `q4.m`

```
function [prime,pseudo2,abs_pseudo] = q4 ( M )

prime = zeros(1,M);
prime(2) = 1;
pseudo2 = zeros(1,M);
abs_pseudo = zeros(1,M);

for N = 3:M
    if euler_test(2,N) == 1
        prime(N) = 1 ;

        p = 2;
        while p < sqrt(N)+1
            if prime(p) == 1
                if mod(N,p) == 0
                    prime(N) = 0 ;
                    pseudo2(N) = 1 ;

                    %%%%%%%%%%% Test for absolute pseudprime %%%%%%%%%%%
```

```
                    abs_pseudo(N) = 1;
                    q = 3;
                    while q < sqrt(N)
                        if prime(q) == 1
                            if mod(N,q) ~= 0
                                if euler_test(q,N) == 0
                                    abs_pseudo(N) = 0;
                                    break
                                else
                                    q = q+1 ;
                                end
                            else
                                q = q+1 ;
                            end
                        else
                            q = q+1;
                        end
                    end
                    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

                    break
                else
                    p = p+1;
                end
            else
                p = p+1;
            end
        end
    end
end
end
```

Listing 12: `q4_2.m`

```
function highest_pseudo_base = q4_2 ( M )

[~,pseudo2,abs_pseudo] = q4 ( M );
q = [ 2 3 5 7 11 13 ];
highest_pseudo_base = 2.*(pseudo2 - abs_pseudo) ;

for N = 1:M
    if highest_pseudo_base(N) == 2
        for i = 2:6
            if euler_test(q(i),N) == 1
                highest_pseudo_base(N) = q(i) ;
            else
                break
            end
        end
    end
end
```

Listing 13: `strong_test.m`

```
function p = strong_test(a,N)

if gcd(a,N) > 1
    p = 0;
    return

else
```

```
    p = 0;
    s = 0;
    t = N - 1;

    while mod(t,2) == 0
        t = t/2;
        s = s +1;
    end

    M = powermod(a,t,N);

    if M == 1 || M == N-1
        p = 1;
        return
    else
        M = mod_mult(M,M,N);
        i = 1;
        while i < s
            if M == 1
                p = 0;
                return
            elseif M == N-1
                p = 1;
                return
            else
                M = mod_mult(M,M,N);
                i = i+1;
            end
        end
    end
end

end
```

Listing 14: `q5.m`

```
function [prime,pseudo2,abs_pseudo] = q5 ( M )

prime = zeros(1,M);
prime(2) = 1;
pseudo2 = zeros(1,M);
abs_pseudo = zeros(1,M);

for N = 3:M
    if strong_test(2,N) == 1
        prime(N) = 1 ;

        p = 2;
        while p < sqrt(N)+1
            if prime(p) == 1
                if mod(N,p) == 0
                    prime(N) = 0 ;
                    pseudo2(N) = 1 ;

                    %%%%%%%%%%% Test for absolute pseudprime %%%%%%%%%%%
                    abs_pseudo(N) = 1;
                    q = 3;
                    while q < sqrt(N)
                        if prime(q) == 1
                            if mod(N,q) ~= 0
                                if strong_test(q,N) == 0
                                    abs_pseudo(N) = 0;
```

```
                                    break
                                else
                                    q = q+1 ;
                                end
                            else
                                q = q+1 ;
                            end
                        else
                            q = q+1;
                        end
                    end
                    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

                    break
                else
                    p = p+1;
                end
            else
                p = p+1;
            end
        end
    end
end
end
```

Listing 15: `q5_2.m`

```
function highest_pseudo = q5_2 ( M )

[~,pseudo2,abs_pseudo] = q5 ( M );

highest_pseudo = 2.*(pseudo2 - abs_pseudo) ;

for N = 1:M
    if highest_pseudo(N) == 2

        if strong_test(3,N) == 1
            highest_pseudo(N) = 3 ;
        else
            break
        end

    end
end
```

Listing 16: `q6.m`

```
[prime,~,~] = q5 ( 10^5 ) ;
fileID1 = fopen('q6_1.csv','w');
fileID2 = fopen('q6_2.csv','w');
fileID3 = fopen('q6_3.csv','w');
fprintf( fileID1 ,' k , sum(primes) , fermat_pseudo2 , euler_pseudo2 , ...
    strong_pseudo2 \n' );
fprintf( fileID2 ,' k , sum(primes) , fermat_pseudo3 , euler_pseudo3 , ...
    strong_pseudo3 \n');
fprintf( fileID3 ,' k , sum(primes) , fermat_pseudo23 , euler_pseudo23 , ...
    strong_pseudo23 \n');

for k = 5:9
    fermat_pseudo2 = 0 ;
    euler_pseudo2  = 0 ;
```

```
12      strong_pseudo2 = 0 ;
13
14      fermat_pseudo3 = 0 ;
15      euler_pseudo3  = 0 ;
16      strong_pseudo3 = 0 ;
17
18      fermat_pseudo23 = 0 ;
19      euler_pseudo23  = 0 ;
20      strong_pseudo23 = 0 ;
21
22      primes = zeros(1,10^5+1);
23
24      for N = 10^k : (10^k + 10^5)
25
26          %%%%%%%%%%%%%%%%%%%%%% Primality Test %%%%%%%%%%%%%%%%%%%%%%%%%%
27          if strong_test(2,N) == 1
28              if strong_test(3,N) == 1
29                  if strong_test(5,N) == 1
30
31                      %%%%%%%%%%%%% Test by trial division %%%%%%%%%%%%%
32                      p = 2 ;
33                      primes(N - 10^k + 1) = 1 ;
34                      while p < sqrt(N) + 1
35                          if prime(p) == 1
36                              if mod(N,p) == 0
37                                  primes(N - 10^k + 1) = 0 ;
38                                  break
39                              else
40                                  p = p + 1 ;
41                              end
42                          else
43                              p = p + 1;
44                          end
45                      end
46                      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
47                  end
48              end
49          end
50          %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
51
52          fermat_pseudo2 = fermat_pseudo2 + fermat_test(2,N) - primes(N - 10^k ...
                + 1);
53          euler_pseudo2  = euler_pseudo2  + euler_test(2,N) - primes(N - 10^k ...
                + 1);
54          strong_pseudo2 = strong_pseudo2 + strong_test(2,N) - primes(N - 10^k ...
                + 1);
55
56          fermat_pseudo3 = fermat_pseudo3 + fermat_test(3,N) - primes(N - 10^k ...
                + 1);
57          euler_pseudo3  = euler_pseudo3  + euler_test(3,N) - primes(N - 10^k ...
                + 1);
58          strong_pseudo3 = strong_pseudo3 + strong_test(3,N) - primes(N - 10^k ...
                + 1);
59
60          fermat_pseudo23 = fermat_pseudo23 + ...
                fermat_test(2,N)*fermat_test(3,N) - primes(N - 10^k + 1);
61          euler_pseudo23  = euler_pseudo23 + euler_test(2,N)*euler_test(3,N) - ...
                primes(N - 10^k + 1);
62          strong_pseudo23 = strong_pseudo23 + ...
                strong_test(2,N)*strong_test(3,N) - primes(N - 10^k + 1);
63      end
64
65      fprintf( fileID1 ,'%d, %d, %d, %d, %d \n', k , sum(primes) , ...
```

```
        fermat_pseudo2 , euler_pseudo2 , strong_pseudo2 );
    fprintf( fileID2 ,'%d, %d, %d, %d, %d \n', k , sum(primes) , ...
        fermat_pseudo3 , euler_pseudo3 , strong_pseudo3 );
    fprintf( fileID3 ,'%d, %d, %d, %d, %d \n', k , sum(primes) , ...
        fermat_pseudo23 , euler_pseudo23 , strong_pseudo23 );

end
```

Listing 17: `q7.m`

```
fprintf('best method \n')
tic

p = zeros(1,10^5) ;
p(1) = 0 ;
p(2) = 1 ;
p(3) = 1 ;

for N = 4:10^5
    if strong_test(2,N) == 1
        if strong_test(3,N) == 1
            p(N) = 1;
        end
    end
end

q = find(p) ;

M = randi([3 10^10],1,10000);
K = zeros(1,length(M));

for i = 1 : length(M)
    if M(i) < 10^5 + 1
        if ismember(M(i),q) == 1
            K(i) = 1 ;
        end

    else
        if strong_test(2,M(i)) == 1
            if strong_test(3,M(i)) == 1
                K(i) = 1;

                j = 1;
                while q(j) < sqrt(M(i))
                    if mod(M(i),q(j)) == 0
                        K(i) = 0 ;
                        break
                    else
                        j = j + 1;
                    end
                end
            end
        end
    end
end
toc

fprintf('trial division \n')

tic
L = zeros(1,length(M));
for i = 1 : length(M)
```

```
    L(i) = prime_td(M(i));
end
toc
```

Listing 18: `q8.m`

```
fileID1 = fopen('q8.csv','w');
fprintf(fileID1,'  t, x, x+y, (100*x)/y \n');
k = 17;
[prime,~,~] = q5 ( 2^k ) ;
for t = 1:5
    for j = 1:3
        strong_prime = ones(1,2^(k-1));

        x = 0;
        y = 0;

        for N = 2.*( 2^(k-2) : 2^(k-1)-1 ) + 1
            if prime(N) == 1
                y = y + 1 ;
            else
                a = randperm(2^(k-1)-1,t);
                for i = 1:t
                    if strong_test(a(i),N) == 0
                        strong_prime(N-2^(k-1)+1) = 0;
                        break
                    end
                end
                x = x + strong_prime(N-2^(k-1)+1) ;
            end
        end
        fprintf(fileID1,' %d, %d, %d, %f \n', t, x, x+y, (100*x)/y);
    end
end
```