1. The code written in C below calculates the sum of the positive integers $a$ and $b$ (written in the decimal system). Suppose that the computer uses the “normal” basic operations (addition, subtraction, multiplication, division,...). Determine whether the algorithm is polynomial or not.

```c
while (b > 0) {
    a = a+1;
    b = b-1;
    printf(''Sum: %d'', a);
}
```

2. The code written in C below calculates the sum of the positive integers $a$ and $b$ (written in the decimal system). Suppose that the computer uses the “normal” basic operations (addition, subtraction, multiplication, division,...). Determine whether the algorithm is polynomial or not. (ceil$(b/2.0)$ gives the upper integer part, and floor$(b/2.0)$ the lower integer part of $\frac{b}{2}$ back.)

```c
while (b > 0) {
    a = a + ceil(b/2.0);
    b = floor(b/2.0);
}
printf(''Sum: %d'', a);
```

3. The code written in C below calculates the largest divisor of $n$ not greater than $a$ (for the integers $0 < a < n$ written in the decimal system). Suppose that the computer uses the “normal” basic operations (addition, subtraction, multiplication, division,...). Determine whether the algorithm is polynomial or not.

```c
while (n % a != 0) {
    a = a-1;
}
printf(''Result: %d'', a);
```

4. The code written in C below calculates $\lfloor \sqrt{n} \rfloor$ (for the positive integer $n$ written in the decimal system). Suppose that the computer uses the “normal” basic operations (addition, subtraction, multiplication, division,...). Determine whether the algorithm is polynomial or not.

```c
x = 0; y = 0;
while (y <= n) {
    x = x+1;
    y = x * x;
}
printf(''Result: %d'', x-1);
```

5. The code written in C below calculates $\lfloor \log_2 n \rfloor$ (for the positive integer $n$ written in the decimal system). Suppose that the computer uses the “normal” basic operations (addition, subtraction, multiplication,...). Determine whether the algorithm is polynomial or not.

```c
x = 0; y = 1;
while (y <= n) {
    x = x+1;
    y = 2 * y;
}
printf(''Result: %d'', x-1);
```
6. (MT’18) The code written in C below calculates the square of the positive integer\( n \) (written in the decimal system). Suppose that the computer uses the “normal” basic operations (addition, subtraction, multiplication, division,...). Determine whether the algorithm is polynomial or not.

```c
x = n; y = 0;
while (x > 0) {
    x = x-1;
    y = y+n;
}
printf(‘‘Result: %d’’, y);
```

7. (MT’18++) The code written in C below calculates the sum of the digits of the positive integer\( n \) (written in the decimal system). Suppose that the computer uses the “normal” basic operations (addition, subtraction, multiplication, division,...). Determine whether the algorithm is polynomial or not. (\( \text{floor}(n/10.0) \) gives the lower integer part of \( n \) back.)

```c
x = 0; y = 0;
while (n > 0) {
    x = \text{floor}(n/10.0);
    y = y+n-10*x;
    n = x;
}
printf(‘‘Result: %d’’, y);
```

8. Calculate the value of the following expressions (you can use a calculator for it, exceptionally):
   a) the remainder of \( 3^{45} \) when divided by 79;
   b) the remainder of \( 5^{300} \) when divided by 623;
   c) the g.c.d. of 673 and 101;
   d) the g.c.d. of 346 and 158;
   e) the integer solutions of the congruence \( 101x \equiv 3 \pmod{673} \);
   f) the integer solutions of the congruence \( 119x \equiv 2 \pmod{514} \);
   g) the integer solutions of the congruence \( 155x \equiv 7 \pmod{352} \).

9. (MT’17+) Use the algorithm we learnt to determine the remainder we get if we divide \( 5^{85} \) by 155.

10. (MT’17++) Let \( n = 123456 \). Use the algorithm we learnt to determine the g.c.d. of \( 12n + 6 \) and \( 9n + 4 \).

11. (MT’18+) Let \( n = 20181210 \). Use the algorithm we learnt to determine the g.c.d. of \( 45n + 12 \) and \( 35n + 9 \).

12. a) Prove that \( 561 = 3 \cdot 11 \cdot 17 \) is a Carmichael number.
    b) Prove that \( 1105 = 5 \cdot 13 \cdot 17 \) is a Carmichael number.

13. We substitute the 26 letters of the alphabet by the numbers 0,1,...,25 (so A = 0, B = 1, C = 2,...,Z = 25). The public key encoding function is \( x \mapsto x^{43} \pmod{85} \). (With this we can encode the numbers 0,1,...,84 but only the first 26 numbers have meaning.) What is the original message if the one encoded by this function is 59 45 7 52 45 75?

14. * (MT’18) Let \( n \) be a positive integer which is divisible by 8 but not divisible by 3. Prove that 3 is a witness of \( n \) (i.e. 3 proves the non-primality of \( n \) in the Fermat test).