HOMEWORK 1
Due: 01.03

1 Integer division

Let $\beta = 2^w$ be the machine word; in the sequel, we shall assume that the processor, given as input $u \in [0, \beta^2 - 1]$ and $v \in [1, \beta - 1]$, can compute $\lfloor u/v \rfloor$ (the integer part of u/v).

Let $A = \sum_{i=0}^{n} a_i\beta^i$ and $B = \sum_{i=0}^{m} b_i\beta^i$ be the expansion of two positive integers in base β. We will prove that the following algorithm returns A/B and $A \mod B$.

Note: in the algorithm INTEGER DIVISION given below the values $(a_i), n$ are assumed to be consistent with the updates of A. That is, whenever the algorithm processes $A \leftarrow$ ..., these values are updated.

```
Integer division
procedure GUESS($A, B$)
    $g \leftarrow \lfloor (a_n\beta + a_{n-1})/b_m \rfloor$
    return $\min(g, \beta - 1)$
end procedure

procedure DIVIDE($A, B$)
    while $A \geq B$ do
        $q_{n-m-1} \leftarrow$ GUESS($A, B$)
        $A \leftarrow A - q_{n-m-1}\beta^{n-m-1}B$
        while $A < 0$ do                                   ▷ Correction loop
            $q_{n-m-1} \leftarrow q_{n-m-1} - 1$
            $A \leftarrow A + B\beta^{n-m-1}$
        end while
    end while
    return $(\sum q_k\beta^k, A)$
end procedure
```

1. Prove that, up to multiplying A and B by suitable powers of 2, we can assume that $b_m \geq \beta/2$

2. Prove that, up to replacing A by $A - \beta^{n-m}B$, we can assume that $A < \beta^{n-m}B$.

   Remark: in the previous two questions, also explain how one should correct quotient and remainder to get the result of the original division in the end...

   Define the following loop invariant (for the outer while loop) ; we denote by A_k, Q_k the value of A, Q after the k-th iteration of the loop (where Q is the integer $\sum_i q_i\beta^i$ for all q_i's that have been defined so far).

- $0 \leq A_k < \beta^{n-m-k}B$
- $A_k + Q_k B = A$.

where n and m are the maximal powers of β that appear in the original A and B (i.e. before the first loop).

3. Deduce that, if the loop invariant is correct, when we exit the loop (Q, A) are the quotient and the remainder of the Euclidean division of A by B.

4. Prove the second statement of the loop invariant, and the non-negativity part of the first one.

   We now turn to proving the loop invariant by induction – to simplify the notations we only deal with the first iteration, but the proof carries over to any iteration. We shall split the proof into two cases: either $g \leq \beta - 1$ in GUESS, or GUESS returns $\beta - 1$.

5. First subcase: $g \leq \beta - 1$. Prove that $q_{n-m-1}b_m \geq a_n\beta + a_{n-1} - b_m + 1$. Deduce that

$$A - q_{n-m-1}B\beta^{n-m-1} \leq \sum_{k=0}^{n-2} a_k\beta^k + (b_m - 1)\beta^{n-1},$$

   and the last part of the loop invariant.

6. Second subcase: $g \geq \beta$. Prove the last part of the loop invariant
   Hint: use question 2.

   We now estimate the number of iterations of the Correction loop.

7. Prove that we always have $\text{GUESS}(A, B)b_m \leq a_n\beta + a_{n-1}$, and deduce that

$$A - \text{GUESS}(A, B)\beta^{n-m-1}B > -\text{GUESS}(A, B)\beta^{n-1}.$$

   Deduce that the number of iterations of the Correction loop is at most 2.
   Hint: use question 1.

8. Conclude on the correction and complexity of the algorithm.

2 Montgomery Multiplication

Let N, R be relatively prime integers. We study an elegant idea of Peter Montgomery on how to compute $\bmod N$ only performing operations $\bmod R$. In particular, when R is a (large) power of 2 and N is odd (and usually large), it gives an efficient algorithm to compute $a^x \bmod N$. In the following, R is a power-of-two.

1. Given N, R relatively prime, let t, s such that $Rs - Nt = 1$. For a natural number a, define the Montgomery reduction as

$$\mathrm{MR}(a) \overset{\text{def}}{=} \frac{1}{R}(a + (at \bmod R) \cdot N). \tag{1}$$

Show that $\mathrm{MR}(a)$ is a non-negative integer. How many operations are needed to compute $\mathrm{MR}()$? Operations here are additions, multiplications, divisions and $\bmod R$.

2. Let b be any integer, $c = at//R$ and $c' = (at - b)//R$, where $//$ denotes integer division. In other words, $c = \lfloor \frac{at}{R} \rfloor$ and $c' = \lfloor \frac{at-b}{R} \rfloor$. Show that

$$\mathrm{MR}(a + bN) = \mathrm{MR}(a) + (c - c')N \tag{2}$$
$$\mathrm{MR}(a + bR) = b + \mathrm{MR}(a) \tag{3}$$

It follows that for $a = 0$, $\mathrm{MR}(bR) = b$ and $\mathrm{MR}(bR^2) = bR$.

3. Let $R > N$ and $R^k = q_k N + r_k$. Show that

$$\mathrm{MR}(ar_k) = aR^{k-1} \bmod N.$$

4. Given $a, b \in \mathbb{Z}$, we want to compute $a \cdot b \bmod N$. Define $a' = a \cdot R \bmod N, b' = b \cdot R \bmod N$, and let

$$a \odot b \overset{\text{def}}{=} MR(ab).$$

First, show that

$$a' \odot b' = aR \odot bR \bmod N.$$

Thus, the operator $\odot$ acts like multiplication $\bmod N$ when restricted to $R\mathbb{Z}$ modulo N.

Second, show that for $a, b < N$ and $R > N$:

$$a \odot b < 2 \cdot N,$$

This means you can avoid taking $\bmod N$ by simply returning $u - N$ when $a \odot b > N$, hence, keeping the size of the output bounded. This is useful when $\odot$ is used iteratively.

5. Now we are ready to develop an algorithm for computing $c = a^x \bmod N$. Let $r = R \bmod N$ and

$$b = \underbrace{a \odot a \odot \ldots \odot a}_{x \text{ times}}.$$

Show that $\mathrm{MR}(br^x)$ returns c.

Give a fast exponentiation algorithm that uses the precomputed values $r = R \bmod N$ and $r_2 = R^2 \bmod N$ (for an appropriately chosen $R > N$) and the $\odot$ operation to compute $a^x \bmod N$. Since r and r_2 are precomputed, the 'on-line' phase of your algorithm should not use $\bmod N$ operations at all. *Hint: the algorithm should be very similar to the 'classical' fast exponentiation algorithm.*