```
def $m, b, p, q$ : int32
def $gen$ : mt19937_64
def $seed$ : uint64

function NEXT_INT()
    $seed \leftarrow seed \oplus seed \gg 12$
    $seed \leftarrow seed \oplus seed \ll 25$
    $seed \leftarrow seed \oplus seed \gg 27$
    return $seed$
end function

function RAND_INT($l, r$)
    return NEXT_INT() $\bmod (r - l + 1) + l$
end function

procedure GENERATE()
    Randomize the Mersenne Twister generator $gen$                ▷ like gen.seed(time(NULL)); in C++
    $seed \leftarrow$ Generate a pseudorandom number from $gen$        ▷ like seed = gen(); in C++
    repeat
        $m \leftarrow$ UNIFORM_INT_DISTRIBUTION($3, 2^{63} - 1, gen$)    ▷ like m = std::uniform_int_distribution<uint64_t>(3, LLONG_MAX)(gen); in C++
    until IS_ODD_PRIME($m$)
    for $i \leftarrow 1$ to 10 do
        NEXT_INT()
    end for
    $b \leftarrow$ RAND_INT($1, m - 1$)
    $p \leftarrow$ RAND_INT($1, m - 1$)
    $q \leftarrow$ RAND_INT($1, m - 1$)
end procedure

function W($c$)
    if $c =$ ( then
        return $p$
    else
        return $q$
    end if
end function

function HASH($s$)
    return $\left(\sum_{i=1}^{|s|} \mathrm{W}(s_i) \cdot b^{i-1}\right) \bmod m$
end function
```