CS 487: Assignment #2 Due: Wed Mar 1, 2023 at 11:59pm

Submission Instructions: Submit your solutions for each question to Crowdmark before the due date and time. Questions 1.(b), 3.(a-d) and 5.(a-d) require you to submit Maple code. A single starter file `Assign2Starter.mpl` is provided to you that you can rename to `FirstLastnameA2.mpl`. You may split this file up into parts for the different questions if you like. Your submission to Crowdmark for the questions that involve Maple coding should include the result of executing the Maple commands, i.e., the plain text file `fileout` produce by running

```
maple < FirstLastnameA2.mpl > out
```

Also email the file `FirstLastnameA2.mpl` to the course account by the due date and time.

1. Consider the following Chinese remaindering problem: given n distinct primes $m_1, m_2, \ldots, m_n$ together with images $v_1, v_2, \ldots, v_n$ with $0 \leq v_i < m_i$, find an integer f such that $0 \leq f < m_1 m_2 \cdots m_n$ and $f \equiv v_i \bmod m_i$ for $1 \leq i \leq n$. As discussed in class, this task can be accomplished in an incremental fashion by computing, for $i = 0, 1, 2, \ldots, n$, the product $M_i = m_1 m_2 \cdots m_i$ together with f_i such that $0 \leq f_i < M_i$ and $f_j \equiv v_j \bmod m_j$ for $1 \leq j \leq i$.

```
$M_0, f_0 := 1, 0$;
for $i$ from 1 to $n$ do
    $M_i, f_i := \mathrm{CRA}(M_{i-1}, m_i, f_{i-1}, v_i)$;
```

   Note that the first return value of function CRA is simply the product of the two moduli Mm_i. The starter file provides a completed implementation of `CRA` called `cra1` that is based on Lagrange interpolation.

   (a) For simplicity, assume that each of the n moduli is one word in length, so that $M_i = m_1 m_2 \cdots m_i$ has word-length i. Assuming the standard algorithms for integer arithmetic, derive a big-O bound in terms of n for the cost of the incremental Chinese remaindering using `cra1`.

   (b) Modify the implementation of `cra2` so that it matches the incremental Chinese remainder algorithm described in class. Note that you should change only a single line in the file. Your answer to this part question should be a printout of a Maple session showing the input and the output.

   (c) Again assuming that each of the n moduli is one word in length, analyze the running time of the incremental Chinese remaindering using your implementation of `cra2`

2. In class we considered the fast multi-point evaluation algorithm, based on the product tree, which will allow a polynomial of degree less than $n = 2^k$ over $\mathsf{R}[x]$ to be evaluated at n distinct points in $O(\mathsf{M}(n) \log n)$ ring operations from R. This last cost estimate remains valid for any multiplication time, in particular also for $\mathsf{M}(n) = O(n(\log n)(\log \log n))$. But now fix $\mathsf{M}(n) = cn^{1+\varepsilon}$, for some constants $\varepsilon, c > 0$, and redo the analysis of the cost of constructing the product tree. Your answer should be in the form of a tight big-O bound for the running time in terms of n and ε.

3. Let $p = 101$, $\mathsf{F} = \mathbb{Z}_{101}$, $f = 30x^7 + 31x^6 + 32x^5 + 33x^4 + 34x^3 + 35x^2 + 36x + 37 \in \mathsf{F}[x]$, $g = 17x^3 + 18x^2 + 19x + 20 \in \mathsf{F}[x]$.

   (a) Compute $\mathrm{rev}(g)^{-1} \bmod x^8$ using Newton iteration. Show the result after each iteration.

   (b) Use (a) and the algorithm based on reversion given in class to find $q, r \in \mathsf{F}[x]$ with $f = qg + r$ and $\deg r < 3$.

   (c) Use the EEA to find $f^{-1} \bmod g$ (that is, find $h \in \mathsf{F}[x]$ with $fh \equiv 1 \bmod g$).

   (d) Now use Newton iteration to find $f^{-1} \bmod g^4$. Show each step of the Newton iteration.

   Your answer to this question should be in the form of a Maple session showing the input and output. Note that you don't need to write maple procedures. Just illustrate the algorithms using a sequence of Maple commands.

4. Let $l \in \mathbb{Z}_{\geq 0}$ and $f = f_0 + f_1 x + \cdots + f_{\ell-1} x^{l-1} \in \mathsf{F}[x]$ with $f_0 \neq 0$ be given. We consider the linear variant of Newton iteration to compute the inverse $g = g_0 + g_1 x + \cdots + g_{l-1} x^{l-1} \in \mathsf{F}[x]$ of f modulo x^ℓ. For $i = 0, 1, \ldots, \ell - 2$, derive an explicit formula for the coefficient g_{i+1} in terms of the coefficients $g_0, g_1, \ldots, g_i$ and the coefficients of the input polynomial f. Analyze this algorithm, and determine the number of operations in F to compute g using the method.

5. In this question, you are to use Maple to trace the algorithm for fast multi-modular reduction and Chinese remaindering. Do not write procedures, simply use Maple commands to compute all the intermediate quantities that are computed by the algorithm for the specific examples that are given. The point is to see how the algorithm works, not just to compute the final result. Consider the list of four moduli $m_0 = 1048583$, $m_1 = 1048589$, $m_2 = 1048601$ and $m_3 = 1048609$.

   (a) Use `igcd` to check that the moduli are pairwise relatively prime.

   (b) Compute the binary tree of products. (I.e., compute the integers at the nodes of the product tree.)

   (c) **Evaluation:** Let $f = 432963076438242135014777$. Compute v_i for $0 \leq i < 4$ such that $0 \leq v_i < m_i$ and $v_i \equiv f \bmod m_i$ by going down the product tree.

   (d) **Chinese Remaindering:**

      i. Compute s_i such that $s_i \equiv (m/m_i)^{-1} \bmod m_i$ for $0 \leq i < 4$. As discussed in class, first compute the $m \bmod m_i^2$ using the product tree approach, then divide by m_i, and finally call `igcdex` to compute the modular inverse.

      ii. Let $v_0 = 751411$, $v_1 = 805076$, $v_2 = 253208$ and $v_3 = 20777$. Compute $c_i = \mathrm{modp}(s_i v_i, m_i)$ for $0 \leq i < 4$, then use the product tree to find an integer a such that $a \equiv v_i \bmod m_i$ for $0 \leq i < 4$. Finally, give the unique integer in the range $[0, m_1 m_2 m_3 m_4 - 1)$ that is congruent to a modulo $m_0 m_1 m_2 m_3$.