Question 1: Factorization in \( \mathbb{Z}[x] \) (30 marks)

Factor the following polynomials in \( \mathbb{Z}[x] \).

\[
p_1 = x^{10} - 6 x^4 + 3 x^2 + 13 \\
p_2 = 8 x^7 + 12 x^6 + 22 x^5 + 25 x^4 + 84 x^3 + 110 x^2 + 54 x + 9 \\
p_3 = 9 x^7 + 6 x^6 - 12 x^5 + 14 x^4 + 15 x^3 + 2 x^2 - 3 x + 14 \\
p_4 = x^{11} + 2 x^{10} + 3 x^9 - 10 x^8 - x^7 - 2 x^6 + 16 x^4 + 26 x^3 + 4 x^2 + 51 x - 170
\]

For each polynomial, first compute its square free factorization. Use the Maple command \texttt{gcd(...)} to do this. Now factor each non-linear square-free factor as follows. Use the Maple command \texttt{Factor(...) mod p} to factor the square-free factors over \( \mathbb{Z}_p \) modulo the primes \( p = 13, 17, 19 \). From this information, determine whether each polynomial is irreducible over \( \mathbb{Z} \) or not. If not irreducible, try to discover what the irreducible factors are by considering combinations of the modular factors and Chinese remaindering (if necessary) and trial division over \( \mathbb{Z} \).

Using Chinese remaindering here is not inefficient in general. Why? Thus for the polynomial \( p_4 \), use Hensel lifting instead. That is, using a suitable prime of your choice from 17, 19, 23, Hensel lift each factor mod \( p \), then determine the irreducible factorization of \( p_4 \) over \( \mathbb{Z} \).

Question 2: Factorization in \( \mathbb{Z}_p[x] \) (30 marks)

(a) Factor the following polynomials over \( \mathbb{Z}_{11} \) using the Cantor-Zassenhaus algorithm.

\[
a_1 = x^4 + 8 x^2 + 6 x + 8, \\
a_2 = x^6 + 3 x^5 - x^4 + 2 x^3 - 3 x + 3, \\
a_3 = x^8 + x^7 + x^6 + 2 x^4 + 5 x^3 + 2 x^2 + 8.
\]

Use Maple to do all polynomial arithmetic, that is, you can use the \texttt{Gcd(...)} \texttt{mod p} and \texttt{Powmod(...)} \texttt{mod p} commands etc., but not \texttt{Factor(...)} \texttt{mod p}.

(b) Compute the square-roots of the integers \( a = 3, 5, 7 \) in the integers modulo \( p \), if they exist, for \( p = 10^{20} + 129 = 1000000000000000129 \) by factoring the polynomial \( x^2 - a \mod p \) using the Cantor-Zassenhaus algorithm. Show your working.
Question 3: A linear $x$-adic Newton iteration (20 marks).

Let $p$ be an odd prime and let $a(x) = a_0 + a_1 x + \ldots + a_n x^n \in \mathbb{Z}_p[x]$ with $a_0 \neq 0$ and $a_n \neq 0$. Suppose $\sqrt{a_0} = \pm u_0 \mod p$. The goal of this question is to design an $x$-adic Newton iteration algorithm that given $u_0$, determines if $u = \sqrt{a(x)} \in \mathbb{Z}_p[x]$ and if so computes $u$. Let

$$u = u_0 + u_1 x + \ldots + u_{k-1} x^{k-1} + \ldots + u_{n-1} x^{n-1}.$$ 

Derive the update formula for $u_k$ given $u^{(k)}$. Show your working.

Now implement your algorithm in Maple and test it on the two polynomials $a_1(x)$ and $a_2(x)$ below using $p = 101$ and $u_0 = +5$. Please print out the sequence of values of $u_0, u_1, u_2, \ldots$ that your program computes. Note, one of the polynomials has a $\sqrt{ }$ in $\mathbb{Z}_p[x]$, the other does not.

$$a_1 = 81 x^6 + 16 x^5 + 24 x^4 + 89 x^3 + 72 x^2 + 41 x + 25$$
$$a_2 = 81 x^6 + 46 x^5 + 34 x^4 + 19 x^3 + 72 x^2 + 41 x + 25$$

Question 4: Cost of the linear $p$-adic Newton iteration (20 marks)

Let $a \in \mathbb{Z}$ and $u = \sqrt{a}$. Suppose $u \in \mathbb{Z}$. The linear $P$-adic Newton iteration for computing $u$ from $u \mod p$ that we gave in class is based on the following linear $p$-adic update formula:

$$u_k = -\frac{\phi_p(f(u^{(k)})/p^k)}{f(u_0)} \mod p.$$ 

where $f(u) = a - u^2$. A direct coding of this update formula for the $\sqrt{ }$ problem in $\mathbb{Z}$ led to the code below where I’ve modified the algorithm to stop if the error $e < 0$ instead of using a lifting bound $B$.

```maple
ZSQRT := proc(a,u0,p) local U,pk,k,e,uk,i;
    u := mods(u0,p);
    i := modp(1/(2*u0),p);
    pk := p;
    for k do
        e := a - u^2;
        if e = 0 then return(u); fi;
        if e < 0 then return(FAIL) fi;
        uk := mods( iquo(e,pk)*i, p );
        u := u + uk*pk;
        pk := p*pk;
    od;
    end:
```

The running time of the algorithm is dominated by the squaring of $u$ in $a := a - u^2$ and the long division of $u$ by $pk$ in `iquo(e,pk)`. Assume the input $a$ is of length $n$ base $p$ digits. At the beginning of iteration $k$, $u = u^{(k)} = u_0 + u_1 p + \ldots + u_{k-1} p^{k-1}$ is an integer of length at most $k$ base $p$ digits. Thus squaring $u$ costs $O(k^2)$ (assuming classical integer arithmetic). In the division of $e$ by $pk = p^k$, $e$ will be an integer of length $n$ base $p$ digits. Assuming classical integer long division is used, this division costs $O((n - k + 1)k)$. Since the loop will run $k = 1, 2, \ldots, n/2$ for the $\sqrt{ }$ problem the total cost of the algorithm is dominated by $\sum_{k=1}^{n/2} k^2 + (n - k + 1)k \in O(n^3)$. 

2
Redesign the algorithm so that the overall time complexity is $O(n^2)$ assuming classical integer arithmetic. Prove that your algorithm is $O(n^2)$. Now implement your algorithm in Maple and verify that it works correctly and that the running time is $O(n^2)$. Use the prime $p = 9973$.

Hint 1: $e = a - u^2 = a - u^{(k)}^2 = a - (u^{(k-1)} + u_{k-1}p^{k-1})^2 = (a - u^{(k-1)}^2) - 2u^{k-1}u_{k-1}p^{k-1} - u_{k-1}^2p^{2k-2}$. Notice that $a - u^{(k-1)}^2$ is the error that was computed in the previous iteration.

Hint 2: We showed that the algorithm for computing the $p$-adic representation of an integer is $O(n^2)$. Notice that it does not divide by $p^k$, rather, it divides by $p$ each time round the loop.