MACM 401/MATH 801
Assignment 5, Spring 2019.

Michael Monagan

Due Friday March 22nd at 4pm. Hand in to dropoff box 1a outside AQ 4100.
Late Penalty: -20% for up to 72 hours late. Zero after that.
For problems involving Maple calculations and Maple programming, you should submit a printout of a Maple worksheet of your Maple session.

Question 1: Factorization in $\mathbb{Z}_p[x]$ (25 marks)

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

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

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

(b) As an application, compute the square-roots of the integers $a = 3, 5, 7$ in the integers modulo p, if they exist, for $p = 10^{20} + 129 = 100000000000000000129$ by factoring the polynomial $x^2 - a$ in $\mathbb{Z}_p[x]$ using the probabilistic factoring algorithm. Show your working. You will have to use `Powmod` here.

For large p, what is the expected time complexity to factor $x^2 - a$ in $\mathbb{Z}_p[x]$ using this probabilistic method? Assume a multiplication in $\mathbb{Z}_p$ costs $O(\log^2 p)$.

Question 2: Factorization in $\mathbb{Z}[x]$ (25 marks)

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

$$a_1 = x^{10} - 6x^4 + 3x^2 + 13$$
$$a_2 = 8x^7 + 12x^6 + 22x^5 + 25x^4 + 84x^3 + 110x^2 + 54x + 9$$
$$a_3 = 9x^7 + 6x^6 - 12x^5 + 14x^4 + 15x^3 + 2x^2 - 3x + 14$$
$$a_4 = x^{11} + 2x^{10} + 3x^9 - 10x^8 - x^7 - 2x^6 + 16x^4 + 26x^3 + 4x^2 + 51x - 170$$

For each polynomial, first compute its square free factorization. You may use the Maple command `gcd(...)` to do this. Now factor each non-linear square-free factor as follows. Use the Maple command `Factor(...) mod p` to factor the square-free factors over $\mathbb{Z}_p$ modulo the primes $p = 13, 17, 19, 23$. 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 efficient in general. Why?
Thus for the polynomial a_4, use Hensel lifting instead. That is, using a suitable prime of your choice from $13, 17, 19, 23$, Hensel lift each factor mod p, then determine the irreducible factorization of a_4 over $\mathbb{Z}$.

Question 3: Cost of the linear p-adic Newton iteration (15 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 $u \bmod p$ that we gave in class is based on the following linear p-adic update formula:

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

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.

```
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, $u = u^{(k)} = u_0 + u_1 p + ... + u_{k-1}p^{k-1}$ is an integer of length at most k base p digits. Thus squaring `u` costs $\mathrm{O}(k^2)$ (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 $k = 1, 2, ..., n/2$ for the $\sqrt{\ }$ problem the total cost of the algorithm is dominated by $\sum_{k=1}^{n/2}(k^2 + (n-k+1)k) \in O(n^3)$.

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 $\mathrm{O}(n^2)$. Use the prime $p = 9973$.

Hint 1: $e = a - (u^{(k)})^2 = a - \left(u^{(k-1)} + u_{k-1}p^{k-1}\right)^2 = (a - (u^{(k-1)})^2) - 2u^{k-1}u_{k-1}p^{k-1} - u_{k-1}^2 p^{2k-2}$.
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 (base p) representation of an integer is $\mathrm{O}(n^2)$. Notice that it does not divide by p^k, rather, it divides by p each time round the loop.

Question 4 (15 marks): Symbolic Integration

Implement a Maple procedure `INT` (you may use `Int` if you prefer) that evaluates antiderivatives $\int f(x)\mathrm{d}x$. For a constant c and positive integer n your Maple procedure should apply

$$\int c\,dx = cx.$$

$$\int cf(x)\,dx \to c\int f(x)\,dx.$$

$$\int f(x) + g(x)\,dx \to \int f(x)\,dx + \int g(x)\,dx.$$

$$\text{For } c \neq 1 \quad \int x^c\,dx = \frac{1}{c+1}x^{c+1}.$$

$$\int x^{-1}\,dx = \ln x.$$

$$\int e^x\,dx = e^x \quad \text{and} \quad \int \ln x\,dx = x\ln x - x.$$

$$\int x^n e^x\,dx \to x^n e^x - \int n x^{n-1} e^x\,dx.$$

$$\int x^n \ln x\,dx \ \text{ by parts.}$$

You may ignore the constant of integration. NOTE: e^x in Maple is `exp(x)`, i.e. it's a function not a power. HINT: use the `diff` command for differentiation to determine if a Maple expression is a constant wrt x. Test your program on the following.

```
> INT( x^2 + 2*x + 1, x );
> INT( x^(-1) + 2*x^(-2) + 3*x^(-1/2), x );
> INT( exp(x) + ln(x) + sin(x), x );
> INT( 2*f(x) + 3*y*x/2 + 3*ln(2), x );
> INT( x^2*exp(x) + 2*x*exp(x), x );
> INT( 2*exp(-x) + ln(2*x+1), x );
> INT( 4*x^3*ln(x) + 3*x^2*ln(x), x );
```

Question 5: 10 marks

Below is some code for the FFT for Assignment 3. The code takes as input an array A and assumes it is indexed from $0..n-1$. It allocates two temporary arrays B and C of size $n/2$ and it overwrites the input A with the output (the input is destroyed).

```
unprotect(FFT);
FFT := proc(n,A,p,w) local n2,B,C,i,wi,T;
   if n=1 then return; fi;
   n2 := n/2;
   B := Array(0..n2-1);
   C := Array(0..n2-1);
   for i from 0 to n2-1 do
       B[i] := A[2*i];
       C[i] := A[2*i+1];
   od;
   FFT(n2,B,p,w^2 mod p);
   FFT(n2,C,p,w^2 mod p);
   wi := 1;
   for i from 0 to n2-1 do
       T := wi*C[i] mod p;
       A[i] := B[i]+T mod p;
       A[n2+i] := B[i]-T mod p;
       wi := w*wi mod p;
   od;
   return;
end:
```

(a) Many of you wrote code which allocates temporary arrays like this. Maple deallocates unused temporary arrays when it does a garbage collection. Allocating and deallocating arrays is not free. It takes time. Let us count the number of arrays allocated. Let $A(n)$ be the number of arrays allocated. Write down a recurrence for the $A(n)$ and initial value and solve it by hand.

(b) How much space is allocated by all these temporary arrays? Let $S(n)$ be the number of words of storage allocated by all the temporary arrays. Assuming an array of size n uses $n+c$ words of storage where the constant c is the number of words to store the size of the array and any other information that Maple needs, and n is for the n entries (integers) in the array. Write down a recurrence relation for $S(n)$ and solve it using Maple's `rsolve` command.

It is possible to redesign the algorithm so that it only needs only one temporary array T of size n. The idea is pass T as an input to the FFT procedure. One can then use the first half of T for the B array and the second half of T for the C array. Then, in the two recursive calls to the FFT, one can use the input array A as temporary space.