```
    |\^/|     Maple 2019 (APPLE UNIVERSAL OSX)
._|\|   |/|_. Copyright (c) Maplesoft, a division of Waterloo Maple Inc. 2019
 \  MAPLE  /  All rights reserved. Maple is a trademark of
 <____ ____>  Waterloo Maple Inc.
      |       Type ? for help.
# Section 10.1
#
# Thm 10.1: Some example of the polynomials x^p - x over Z/(p).
#
> for i to 3 do
>    p := ithprime(i);
>    print(p,Factor(x^p-x) mod p);
> od:
                                 2, (x + 1) x

                             3, (x + 1) x (x + 2)

                     5, (x + 4) (x + 1) x (x + 3) (x + 2)


##########################################################################
#
# Thm 10.2: An example of the polynomial x^(p^d) - x over Z/(p).
#
# Consider p = 3 and d = 4.
#
> Factor(x^(3^4)-x) mod 3;
  4            4    3    2        4    2              4    3    2
(x  + x + 2) (x  + x  + x  + 1) (x  + x  + 2 x + 1) (x  + x  + x  + 2 x + 2)

    4      3    2        2              4      3
   (x  + 2 x  + x  + 1) (x  + 2 x + 2) (x  + 2 x  + x + 1) (x + 1)

    4      3      2           4              2            2
   (x  + 2 x  + 2 x  + x + 2) (x  + 2 x + 2) (x  + x + 2) (x  + 1)

    4      3    2             4    3      2              4    3
   (x  + 2 x  + x  + x + 2) (x  + x  + 2 x  + 2 x + 2) (x  + x  + 2 x + 1) x

    4    2        4      2        4    3    2             4      3
   (x  + x  + 2) (x  + 2 x  + 2) (x  + x  + x  + x + 1) (x  + 2 x  + 2)
```

```
   4    2            4      3    2                    4    3
 (x  + x  + x + 1) (x  + 2 x  + x  + 2 x + 1) (x + 2) (x  + x  + 2)


# We know that x^(3^4) - x is the product of all irreducible of
# degree 1, 2 and 4.  From inspecting the factorization above we
# can see that there are 18 irreducibles of degree 4 over Z/(3)[x].

########################################################################
#
# But note that we can directly enumerate the number of distinct
# irreducibles of degree d over Z/(p), without factoring.
#
# Question: How many distinct irreducibles of degree i are there over Z/(3)
#           for i = 1,2,3,4?
#
# Derivation:
#
# Let ki be the number of distinct irreducibles of degree i over Z/(3).
# Then we can set up a system of linear equations.
#
> sys := {degree(x^3-x,x) = k1*1,                    # 1 divisible by 1
>         degree(x^(3^2)-x,x) = k1*1 + k2*2,         # 2 divisible by 1, 2
>         degree(x^(3^3)-x,x) = k1*1 + k3*3,         # 3 divisible by 1, 3
>         degree(x^(3^4)-x,x) = k1*1 + k2*2 + k4*4}: # 4 divisible by 1, 2, 4

> vars := {k1,k2,k3,k4};
                            vars := {k1, k2, k3, k4}


> sys;
         {3 = k1, 9 = k1 + 2 k2, 27 = k1 + 3 k3, 81 = k1 + 2 k2 + 4 k4}


> solve(sys,vars);
                       {k1 = 3, k2 = 3, k3 = 8, k4 = 18}



########################################################################
#
# As an aside, it is easy to derive the following function.  In Maple,
```

```
# "option remember" stores the results of previous function calls so they
# don't need to be recomputed (or manually stored in a table like
# dynamic programming).
#
# Input: p - a prime
#        d - an integer in Z_{>=1}
#
# Output: the number of distinct irreducibles of degree d over Z/(p)[x]
#
> foo := proc(p,d)
>     option remember;
>     local i,c;

>     if d=1 then return p fi; # the degree of x^p - x

>     # compute sum of degrees of all irreducibles in x^(p^d) - x of deg < d
>     c := 0;
>     for i to d-1 do                  # degree i   # no. of irred. of degree i
>        if modp(d,i)=0 then c := c + i          * foo(p,i) fi
>     od;

>     return iquo(p^d - c,d);

> end:

> foo(3,4);
                                        18

#
# Some more checks that foo is correct:
#
> degree(x^(5^4) - x,x) = foo(5,1)*1 + foo(5,2)*2 + foo(5,4)*4;
                                    625 = 625

> degree(x^(17^26) - x,x) = foo(17,1)*1 + foo(17,2)*2 + foo(17,26)*26;
      98100666009922840441972689847969 = 98100666009922830537394656942049


#########################################################################
#
# A key computational tool (for efficiency) that is used in the factoring
# algorithm is binary powering modulo another polynomial.  This is
```

```
# algorithm "RepeatedSquaring" in the script.
#
# One possible implementation is in the posted example "mypowmod.mpl".
#
# This operation is important enough that Maple has a built-in function
# for it, namely Powmod.
#
# Here is an example of an application of Powmod.
#
# Generate a rather large degree random polynomial with many factors
# modulo 3.
#
> f := mul(Randpoly(i,x)^i mod 3,i=1..60): f := Expand(f) mod 3:
#
# Let's not print out f! But do look at its degree.
#
> degree(f,x);
                                     73810

#
# Compute the product of all irreducibles of degree 1 that divide f.
# (One copy of each).
#
> g1 := Gcd(x^3-x,f) mod 3;
                                        3
                                 g1 := x  + 2 x

#
# Check that f1 has only linear factors:
#
> Factor(g1) mod 3;
                               (x + 1) x (x + 2)

#
# To obtain all irreducibles of f of degree 1 (with multiplicity) we
# can compute the gcd of f with a high power of (x^3 - x).  Since
# a linear factor can divide f at most deg f times, it suffices to compute
#
> g1_with_multiplicities := Gcd((x^3-x)^degree(f,x),f) mod 3:
> Factor(%) mod 3;
                                 704  979        1119
                          (x + 1)    x    (x + 2)
```

```
#
# Instead of first computing (x^3 - x)^degree(f,x) and then taking the
# gcd, it is more efficient (polynomial time vs. exponential time!) to
# compute Rem((x^3-x)^degree(f,x),f) using repeated squaring and then
# take the gcd:
#
> Gcd(Powmod((x^3-x),degree(f,x),f,x), f) mod 3:
> Factor(%) mod 3;
                                704  979        1119
                         (x + 1)    x    (x + 2)

#
# The above idea is used in the posted example "DDFact.mpl" which gives
# a function to compute the distinct degree factorization of the
# the squarefree part of f (even if f itself is not squarefree).
# Note: To improve the efficiency of that routine the computation
# g^ceil(n/i) should be replaced with the appropriate call to Powmod.

#########################################################################
#
# Section 10.2
#
# Let us first illustrate Thm 10.3.
#
# We know that R = Z/(p) for p a prime is finite field with p element,
# in particular it is simply the set {0,1,...,p-1} with addition and
# multiplication modulo p.
#
# The first part of Thm 10.3 states that if h is an irreducible of
# degree d then R[x]/<h> is finite field with p^d elements.
#
# As an example, consider p = 3 (so R = Z/(3)) and d = 2.
#
> h := x^2 + 1;  # irred. of degree 2 over Z/(3)
                                        2
                                  h := x  + 1

#
# Then RR = R[x]/<h> is a finite field with p^d = 3^2 = 9 elements.
# The elements of RR is all polynomials over R[x] of degree
# strictly less than 2 (i.e., the distinct polynomials of R[x] modulo h).
```

```
#
> RR := {seq(seq(i*x + j,i=0..2),j=0..2)};
          RR := {0, 1, 2, x, 2 x, x + 1, x + 2, 2 x + 1, 2 x + 2}

#
# Addition/subtraction in R is just addition/subtraction modulo 3.
# Multiplication is done modulo h, e.g., the product of (x+2) * (2*x+1) is
#
> Rem((x+2)*(2*x+1),h,x) mod 3;
                                     2 x

#
# The second part of Thm 10.3 should be familiar from our study
# of the Chinese remainder theorem, and algorithm such as multi-modular
# reduction.

##########################################################################
#
# Now consider Thm 10.4.
#
# First consider a prime Z/(p).  The theorem says that any nonzero
# element of Z/(p), when raised to the power (p-1)/2, will be equal
# to 1 or -1, with exactly half equal to 1.  Some experimental
# confirmation of the theorem.
#
> for i to 5 do
>    p := ithprime(i);
>    S := [seq(a,a=1..p-1)]; # nonzero elements of Z/(p)
>    R := map(a->mods(a^((p-1)/2),p),S);
>    print(p,S,R);
> od:
                                2, [1], [1]

                             3, [1, 2], [1, -1]

                       5, [1, 2, 3, 4], [1, -1, -1, 1]

                 7, [1, 2, 3, 4, 5, 6], [1, 1, -1, 1, -1, -1]

    11, [1, 2, 3, 4, 5, 6, 7, 8, 9, 10], [1, -1, 1, 1, 1, -1, -1, -1, 1, -1]

#
```

```
# Obviously, if we select an element of {1,2,...,p-1} uniformly
# at random, then with probably 1/2 it will be a quadratic residue
# and with probability 1-1/2=1/2 it will not be a non-quadratic residue.
#
##########################################################################
#
# Now let's give an illustration of the equal degree factorization
# on page 4 of the script.  Let R = Z/(p), p = 10000019.
#
> p := 10000019;
                                 p := 10000019

> d := 2;
                                     d := 2

> h1 := x^2+2090578*x+4297752: # irred. of degree 2
> h2 := x^2+4958404*x+3788058: # irred. of degree 2
> f := Expand(h1*h2) mod p;
                  4           3            2
            f := x  + 7048982 x  + 8708093 x  + 5889928 x + 2913426

#
# Our goal is to factor f over R[x].  (Pretend we don't know h1 and h2.)
#
# The residue class ring
#
#            R/<f> \equiv R/<h1> x R/<h2>   (*)
#
# contains (p)^2 elements, of which (p-1)^2 are relatively prime to f. (Why?)
#
# Four of the elements that are relatively prime to f are
# S = { (1,1),(1,-1),(-1,1),(-1,-1) }: our goal is to uniformly and randomly
# select one of these four elements.
# On the left hand side of (*) the elements of S corresond to {e1,e2,e2,e4}:
#
> Gcdex(h1,h2,x,'s','t') mod p;
                                       1

> e1 := Rem((1)*s*h1 + (1)*t*h2,f,x) mod p;
                                    e1 := 1

> e2 := Rem((-1)*s*h1 + (1)*t*h2,f,x) mod p;
```

```
                              3            2
              e2 := 2017226 x  + 4937057 x  + 544669 x + 3886491

> e3 := Rem((1)*s*h1 + (-1)*t*h2,f,x) mod p;
                             3            2
             e3 := 7982793 x  + 5062962 x  + 9455350 x + 6113528

> e4 := Rem((-1)*s*h1 + (-1)*t*h2,f,x) mod p;
                                 e4 := 10000018

#
# Note that subtracting (1,1) from the elements in S gives the set
# { (0,0), (0,-2), (-2,0), (-2,-2) }: some of these (in this case
# two) have the nice property that they "split" the polynomial f:
#
> Gcd(e1-1,f) mod p; # should give us f (useless)
                4            3            2
               x  + 7048982 x  + 8708093 x  + 5889928 x + 2913426

> Gcd(e2-1,f) mod p; # should give us h1 (great!)
                             2
                            x  + 2090578 x + 4297752

> Gcd(e3-1,f) mod p; # should give us h2 (great!)
                             2
                            x  + 4958404 x + 3788058

> Gcd(e4-1,f) mod p; # should give us 1 (useless)
                                       1

#
# Our goal is to select an element from {e1,e2,e2,e4) uniformly at random.
# To do so, we make use of Thm 10.4.
# First choose a random polynomial of degree < 4.
#
> u := Randpoly(4,x) mod p; # returns a random polynomial of degree 4
                    4            3            2
         u := 10323 x  + 1265632 x  + 4335155 x  + 3429475 x + 2575038

> u := u - lcoeff(u,x)*x^4; # now we have a random polynomial of degree < 4
                             3            2
               u := 1265632 x  + 4335155 x  + 3429475 x + 2575038
```

```
#
# We can check that u is relatively prime to f using Gcd.
#
> Gcd(f,u) mod p;
                                       1

#
# [Excercise: What is the chance that our u is not relatively prime to f?]
# [Question: What if u is not realtively prime to f?  Is this bad?]
#
# Assume now that u is relatively prime to f.
# Use Powmod to raise u to the power (p^d-1)/2.
#
> uu := Powmod(u,(p^2-1)/2,f,x) mod p;
                                 uu := 10000018

#
# Then we must have uu in {e1,e2,e2,e4}.
# Let's try to take the gdd of uu - 1 with f:
#
> Gcd(uu-1,f) mod p;
                                       1

#
# Unlucky!  Try again.
#
> uu := Powmod(u,(p^2-1)/2,f,x) mod p;
                                 uu := 10000018

> Gcd(uu-1,f) mod p;
                                       1

#
# Now we have split f.
> quit
memory used=117.5MB, alloc=32.6MB, time=41.22
```