

```
> restart;
```

Mod Arithmetic

Checking if a List is a Permutation

This procedure tests if a given list of numbers is a permutation, i.e. all entries are different from each other. In particular, if the length of the list is m and all members are integers in the range $0, \dots, m-1$ (as occurs when doing arithmetic mod m), then the list is a permutation iff it is a permutation of $0, \dots, m-1$.

```
> permutation := proc(num_array)
  local i, j, perm, m :
  m := nops(num_array) : perm := true :
  for i from 1 to m-1 do
    for j from i+1 to m do
      if num_array[i]=num_array[j] then perm := false: fi
    od od:
  perm;
end:
```

Here is an example.

```
> permutation([2,3,1]); permutation([1,1,6,1,6,6]);
                                     true
                                     false
```

Multiples of $\{1, \dots, n-1\}$ by a fixed factor, computed mod n

This procedure produces, for $W = 1, \dots, n-1$, a list of all multiples $W*r \text{ mod } n$.

```
> mod_mult := proc(r,n)
  local mult, W :
  for W from 1 to n-1 do mult[W]:= W*r mod n od :
  convert(mult, 'list');
end:
```

Here is an example.

```
> mod_mult(3,6) ; mod_mult(5,8);
                                     [3, 0, 3, 0, 3]
                                     [5, 2, 7, 4, 1, 6, 3]

> permutation(mod_mult(3,6)); permutation(mod_mult(5,8));
                                     false
                                     true
```

Powers of $\{1, \dots, n-1\}$ by a fixed exponent, computed mod n

This procedure produces, for $W = 1, \dots, n-1$, a list of all powers $W^e \text{ mod } n$.

```
> mod_power := proc(e,n)
```

```

local power, W :
for W from 1 to n-1 do power[W]:= W&^e mod n   od :
convert(power, 'list');
end:

```

Here are some examples.

```

> mod_power(3,7) ; mod_power(17,33);
                                     [1, 1, 6, 1, 6, 6]
[1, 29, 9, 16, 14, 30, 28, 2, 15, 10, 11, 12, 7, 20, 27, 25, 8, 6, 13, 26, 21, 22, 23, 18, 31, 5, 3, 19, 17, 24, 4,
 32]

```

— Powers of members of any list by a fixed exponent, computed mod n

This procedure produces, for any list $L = \{a,b, \dots\}$ and exponent d , a list of the powers (a^d, b^d, \dots) computed mod n .

```

> mod_power_list := proc(L,d,n)
  local power, i :
  for i from 1 to nops(L) do power[i]:= L[i]&^d mod n   od :
  convert(power, 'list');
end:

```

Here is an example.

```

> mod_power_list([2,5,8],3,5);
                                     [3, 0, 2]

```

— A Property of Mod Multiplication

This example demonstrates a property of mod multiplication.

It first checks if r and n are relatively prime.

If so, it shows that for the multiplication table mod n , the numbers in the row corresponding to multiplication by r each occur exactly once.

In other words, multiplication by r mod n produces a permutation (shuffle) of $\{1, \dots, n-1\}$. Moreover, multiplication by the inverse s of r mod n reverse shuffles back to the original order.

(Try some other examples for yourself.)

```

> n := 25; r := 6;
                                     n := 25
                                     r := 6

> mod_mult(1,n);
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24]

> gcd(r,n);
                                     1

> mod_mult(r,n);
[6, 12, 18, 24, 5, 11, 17, 23, 4, 10, 16, 22, 3, 9, 15, 21, 2, 8, 14, 20, 1, 7, 13, 19]

```

```

> permutation(mod_mult(r,n));
                                     true

> s := 1/r mod n;
                                     s := 21

> s * mod_mult(r,n) mod n;
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24]

```

— Fermat's Little theorem

This is an example of Fermat's Little Theorem.

(Also try some other examples of your own.)

```

> mod_power(7,7); mod_power(6,7);
                                     [1, 2, 3, 4, 5, 6]
                                     [1, 1, 1, 1, 1, 1]

```

Here is another example.

```

> mod_power(31,31); mod_power(30,31);
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]

```

— RSA Cryptography and a Property of Mod Powers

This example looks at the type of powers $W^e \bmod n$ that occur in RSA cryptography.

We suppose that p and q are distinct primes, and let $n = p \cdot q$, $m = (p-1) \cdot (q-1)$.

The example checks if e and m are relatively prime.

If they are, it then shows that taking powers $W^e \bmod n$ of W produces a permutation (shuffle) of $\{1, \dots, n-2\}$, and that taking powers again by the inverse d of $e \bmod m$ reverse shuffles back to the original order.

(Try some examples of your own.)

In RSA cryptography, p and q are random primes (MUCH bigger than the following example).

```

> p:=17; isprime(p); q:=5 ; isprime(q);
                                     p := 17
                                     true
                                     q := 5
                                     true

> n := p*q; m := (p-1)*(q-1);
                                     n := 85
                                     m := 64

```

The number e is the encoding exponent.

```

> e := 27; gcd(e,m);

```

```
e := 27
```

```
1
```

In RSA cryptography, and in this "toy" example, the secret number W is one of the following (but not too small).

```
> W = mod_power(1, n);
```

```
W = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84]
```

The secret number W is raised to the power e to give the corresponding coded number C as in the following list.

```
> C = mod_power(e, n);
```

```
C = [1, 8, 7, 64, 45, 56, 48, 2, 49, 20, 46, 23, 72, 44, 60, 16, 68, 52, 59, 75, 81, 28, 22, 14, 70, 66, 3, 12, 74, 55, 61, 43, 67, 34, 35, 76, 58, 47, 79, 5, 31, 53, 32, 54, 80, 6, 38, 27, 9, 50, 51, 18, 42, 24, 30, 11, 73, 82, 19, 15, 71, 63, 57, 4, 10, 26, 33, 17, 69, 25, 41, 13, 62, 39, 65, 36, 83, 37, 29, 40, 21, 78, 77, 84]
```

We check that this operation does indeed "shuffle" the set of all possible numbers.

```
> permutation(mod_power(e, n));
```

```
true
```

The number d is the decoding exponent.

```
> d := 1/e mod m;
```

```
d := 19
```

Raising the coded number C to the power d gives the decoded number D .

This should be the same as the original secret number W , which we now see it is!

```
> D = mod_power_list(mod_power(e, n), d, n);
```

```
D = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84]
```

```
>
```