Package ‘Zseq’
February 3, 2018

Type Package
Title Integer Sequence Generator
Version 0.2.0
License GPL (>= 3)
Encoding UTF-8
LazyData true
Imports gmp
RoxygenNote 6.0.1
NeedsCompilation no
Author Kisung You [aut, cre] (<https://orcid.org/0000-0002-8584-459X>)
Maintainer Kisung You <kyou@nd.edu>
Repository CRAN
Date/Publication 2018-02-02 23:07:14 UTC

R topics documented:

Zseq-package .................................................. 2
Abundant .......................................................... 3
Achilles .......................................................... 3
Bell .............................................................. 4
Carmichael ...................................................... 5
Catalan .......................................................... 5
Composite ....................................................... 6
Deficient ......................................................... 7
Equidigital ....................................................... 7
Evil ............................................................. 8
Extravagant ...................................................... 9
Description

The world of integer sequence has long history, which has been accumulated in OEIS. Even though R is not a first pick for many number theorists, we introduce our package to enrich the R ecosystem as well as provide pedagogical toolset. We adopted gmp for flexible large number computations in that users can easily experience large number sequences on a non-exclusive generic computing platform.
Abundant

**Abundant numbers**

**Description**

Under OEIS A005101, an abundant number is a number whose proper divisors sum up to the extent greater than the number itself. First 6 abundant numbers are 12, 18, 20, 24, 30, 36.

**Usage**

\[
\text{Abundant}(n, \ gmp = \text{TRUE})
\]

**Arguments**

- **n** the number of first \( n \) entries from the sequence.
- **gmp** a logical; \text{TRUE} to use large number representation, \text{FALSE} otherwise.

**Value**

a vector of length \( n \) containing first entries from the sequence.

**See Also**

Deficient, Perfect

**Examples**

\[
\text{CC generate first 30 Abundant numbers and print it}
\]

\[
\text{print(Abundant(30))}
\]

Achilles

**Achilles numbers**

**Description**

Under OEIS A052486, an Achilles number is a number that is powerful but not perfect. First 6 Achilles numbers are 72, 108, 200, 288, 392, 432.

**Usage**

\[
\text{Achilles}(n, \ gmp = \text{TRUE})
\]

**Arguments**

- **n** the number of first \( n \) entries from the sequence.
- **gmp** a logical; \text{TRUE} to use large number representation, \text{FALSE} otherwise.
Bell Value

a vector of length \( n \) containing first entries from the sequence.

Examples

```r
## generate first 3 Achilles numbers and print
print(Achilles(3))
```

---

Bell numbers

Description

Under OEIS A000110, the \( n \)th Bell number is the number of ways to partition a set of \( n \) labeled elements, where the first 6 entries are 1, 1, 2, 5, 15, 52.

Usage

```r
Bell(n, gmp = TRUE)
```

Arguments

- \( n \) the number of first \( n \) entries from the sequence.
- \( gmp \) a logical; TRUE to use large number representation, FALSE otherwise.

Value

a vector of length \( n \) containing first entries from the sequence.

Examples

```r
## generate first 30 Bell numbers and print
print(Bell(30))
```
Carmichael numbers

Description
Under OEIS A002997, a Carmichael number is a composite number $n$ such that

$$b^{n-1} = 1 \pmod{n}$$

for all integers $b$ which are relatively prime to $n$. First 6 Carmichael numbers are 561, 1105, 1729, 2465, 2821, 6601.

Usage
Carmichael(n, gmp = TRUE)

Arguments
- $n$: the number of first $n$ entries from the sequence.
- gmp: a logical; TRUE to use large number representation, FALSE otherwise.

Value
a vector of length $n$ containing first entries from the sequence.

Examples
```r
## generate first 3 Carmichael numbers
print(Carmichael(3))
```

Catalan numbers

Description
Under OEIS A000108, the $n$th Catalan number is given as

$$C_n = \frac{(2n)!}{(n+1)!n!}$$

where the first 6 entries are 1, 1, 2, 5, 14, 42 with $n \geq 0$.

Usage
Catalan(n, gmp = TRUE)
**Arguments**

- **n**  
  the number of first \( n \) entries from the sequence.

- **gmp**  
  a logical; TRUE to use large number representation, FALSE otherwise.

**Value**

a vector of length \( n \) containing first entries from the sequence.

**Examples**

```r
## generate first 30 Catalan numbers
print(Catalan(30))
```

---

**Composite**

**Composite numbers**

**Description**

Under OEIS A002808, a *composite* number is a positive integer that can be represented as multiplication of two smaller positive integers. The first 6 composite numbers are 4, 6, 8, 9, 10, 12.

**Usage**

```r
Composite(n, gmp = TRUE)
```

**Arguments**

- **n**  
  the number of first \( n \) entries from the sequence.

- **gmp**  
  a logical; TRUE to use large number representation, FALSE otherwise.

**Value**

a vector of length \( n \) containing first entries from the sequence.

**Examples**

```r
## generate first 30 Composite numbers
print(Composite(30))
```
Deficient

Deficient numbers

Description

Under OEIS A005100, a deficient number is a number whose proper divisors sum up to the extent smaller than the number itself. First 6 abundant numbers are 1, 2, 3, 4, 5, 7

Usage

Deficient(n, gmp = TRUE)

Arguments

n the number of first n entries from the sequence.

gmp a logical; TRUE to use large number representation, FALSE otherwise.

Value

a vector of length n containing first entries from the sequence.

See Also

Abundant, Perfect

Examples

## generate first 30 Deficient numbers
print(Deficient(30))

---

Equidigital

Equidigital numbers

Description

Under OEIS A046758, an Equidigital number has equal digits as the number of digits in its prime factorization including exponents. First 6 Equidigital numbers are 1, 2, 3, 5, 7, 10. Though it doesn’t matter which base we use, here we adopt only a base of 10.

Usage

Equidigital(n, gmp = TRUE)
Arguments

- **n**: the number of first \( n \) entries from the sequence.
- **gmp**: a logical; TRUE to use large number representation, FALSE otherwise.

Value

- a vector of length \( n \) containing first entries from the sequence.

See Also

- Frugal, Extravagant

Examples

```r
# generate first 20 Equidigital numbers
print(Equidigital(20))
```

---

**Evil**

**Evil numbers**

Description

Under OEIS A001969, an *Evil* number has an even number of 1’s in its binary expansion. First 6 Evil numbers are 0, 3, 5, 6, 9, 10.

Usage

```r
Evil(n, gmp = TRUE)
```

Arguments

- **n**: the number of first \( n \) entries from the sequence.
- **gmp**: a logical; TRUE to use large number representation, FALSE otherwise.

Value

- a vector of length \( n \) containing first entries from the sequence.

See Also

- Odious

Examples

```r
# generate first 20 Evil numbers
print(Evil(20))
```
Extravagant numbers

Description

Under OEIS A046760, an Extravagant number has less digits than the number of digits in its prime factorization including exponents. First 6 Extravagant numbers are 4, 6, 8, 9, 12, 18. Though it doesn’t matter which base we use, here we adopt only a base of 10.

Usage

\[ \text{Extravagant}(n, \text{gmp} = \text{TRUE}) \]

Arguments

- \( n \) the number of first \( n \) entries from the sequence.
- \( \text{gmp} \) a logical; TRUE to use large number representation, FALSE otherwise.

Value

A vector of length \( n \) containing first entries from the sequence.

See Also

Frugal, Equidigital

Examples

```R
## generate first 20 Extravagant numbers
print(Extravagant(20))
```

Factorial numbers

Description

Under OEIS A000142, a Factorial is the product of all positive integers smaller than or equal to the number. First 6 such numbers are 1, 1, 2, 6, 24, 120

Usage

\[ \text{Factorial}(n, \text{gmp} = \text{TRUE}) \]
Arguments

\( n \)  
the number of first \( n \) entries from the sequence.

\( gmp \)  
a logical; TRUE to use large number representation, FALSE otherwise.

Value

a vector of length \( n \) containing first entries from the sequence.

Examples

```r
## generate first 10 Factorials
print(Factorial(10))
```

Description

Under OEIS A005165, an Alternating Factorial is the absolute value of the alternating sum of the first \( n \) factorials of positive integers. First 6 such numbers are 0, 1, 1, 5, 19, 101.

Usage

```r
Factorial.Alternating(n, gmp = TRUE)
```

Arguments

\( n \)  
the number of first \( n \) entries from the sequence.

\( gmp \)  
a logical; TRUE to use large number representation, FALSE otherwise.

Value

a vector of length \( n \) containing first entries from the sequence.

See Also

Factorial

Examples

```r
## generate first 5 Alternating Factorial numbers
print(Factorial.Alternating(5))
```
Factorial.Double

Double Factorial numbers

Description
Under OEIS A000165 and A001147, a Double Factorial is the factorial of numbers with same parity. For example, if \( n = 5 \), then \( n!! = 5 \ast 3 \ast 1 \).

Usage
Factorial.Double(n, gmp = TRUE, odd = TRUE)

Arguments
- \( n \): the number of first \( n \) entries from the sequence.
- \( gmp \): a logical; TRUE to use large number representation, FALSE otherwise.
- \( odd \): a logical; TRUE for double factorial of odd numbers, FALSE for even numbers.

Value
a vector of length \( n \) containing first entries from the sequence.

See Also
Factorial

Examples
```r
## generate first 10 odd Factorials
print(Factorial(10))
```

Fibonacci

Fibonacci numbers

Description
Under OEIS A000045, the \( n \)th Fibonacci number is given as

\[
F_n = F_{n-1} + F_{n-2}
\]

where the first 6 entries are 0, 1, 1, 2, 3, 5 with \( n \geq 0 \).

Usage
Fibonacci(n, gmp = FALSE)
Frugal

Arguments

- **n**: the number of first \( n \) entries from the sequence.
- **gmp**: a logical; TRUE to use large number representation, FALSE otherwise.

Value

- a vector of length \( n \) containing first entries from the sequence.

Examples

```r
## generate first 30 Fibonacci numbers
print(Fibonacci(30))
```

---

**Description**

Under OEIS A046759, a *Frugal* number has more digits than the number of digits in its prime factorization including exponents. First 6 Frugal numbers are 125, 128, 243, 256, 343, 512. Though it doesn’t matter which base we use, here we adopt only a base of 10.

**Usage**

```r
Frugal(n, gmp = TRUE)
```

Arguments

- **n**: the number of first \( n \) entries from the sequence.
- **gmp**: a logical; TRUE to use large number representation, FALSE otherwise.

Value

- a vector of length \( n \) containing first entries from the sequence.

See Also

*Extravagant*, *Equidigital*

Examples

```r
## generate first 5 Frugal numbers
print(Frugal(5))
```
**Happy**  

**Happy numbers**

**Description**

Under OEIS A007770, a *Happy* number is defined by the process that starts from arbitrary positive integer and replaces the number by the sum of the squares of each digit until the number is 1. First 6 Happy numbers are 1, 7, 10, 13, 19, 23.

**Usage**

```r
Happy(n, gmp = TRUE)
```

**Arguments**

- `n` the number of first `n` entries from the sequence.
- `gmp` a logical; `TRUE` to use large number representation, `FALSE` otherwise.

**Value**

a vector of length `n` containing first entries from the sequence.

**Examples**

```r
# generate first 30 happy numbers
print(Happy(30))
```

**Juggler**  

**Juggler sequence**

**Description**

Under OEIS A094683, a *Juggler* sequence is an integer-valued sequence that starts with a nonnegative number iteratively follows that \( J_{k+1} = \text{floor}(J_{k}^{1/2}) \) if \( J_{k} \) is even, or \( J_{k+1} = \text{floor}(J_{k}^{3/2}) \) if odd. No first 6 terms are given since it all depends on the starting value.

**Usage**

```r
Juggler(start, gmp = TRUE)
```

**Arguments**

- `start` the starting nonnegative integer.
- `gmp` a logical; `TRUE` to use large number representation, `FALSE` otherwise.
Value

a vector recording the sequence of unknown length a priori.

Examples

```r
# let's start from 9 and show the sequence
print(Juggler(9))
```

---

**Juggler.Largest**  
**Largest value for Juggler sequence**

Description

Under OEIS [A094716](https://oeis.org/A094716), the *Largest value for Juggler sequence* is the largest value in trajectory of a sequence that starts from \( n \). First 6 terms are 0, 1, 2, 36, 4, 36 that \( n \) starting from 0 is conventional choice.

Usage

```r
Juggler.Largest(n, gmp = TRUE)
```

Arguments

- **n**: the number of first \( n \) entries from the sequence.
- **gmp**: a logical; TRUE to use large number representation, FALSE otherwise.

Value

a vector of length \( n \) containing first entries from the sequence.

See Also

- [Juggler](#)

Examples

```r
# generate first 10 numbers of largest values for Juggler sequences
print(Juggler.Largest(10))
```
### Juggler.Nsteps

**Number of steps for Juggler sequence**

**Description**

Under OEIS A007320, a *Number of steps for Juggler sequence* literally counts the number of steps required for a sequence that starts from \( n \). First 6 terms are 0, 1, 6, 2, 5, 2 that \( n \) starting from 0 is conventional choice. Note that when it counts *number of steps*, not the length of the sequence including the last 1.

**Usage**

\[
\text{Juggler.Nsteps}(n, \text{gmp} = \text{TRUE})
\]

**Arguments**

- **n**
  - the number of first \( n \) entries from the sequence.
- **gmp**
  - a logical; TRUE to use large number representation, FALSE otherwise.

**Value**

- a vector of length \( n \) containing first entries from the sequence.

**See Also**

- [Juggler](#)

**Examples**

```r
## generate first 10 numbers of steps for Juggler sequences
print(Juggler.Nsteps(10))
```

### Lucas

**Lucas numbers**

**Description**

Under OEIS A000032, the \( n \)th *Lucas* number is given as

\[
F_n = F_{n-1} + F_{n-2}
\]

where the first 6 entries are 2, 1, 3, 4, 7, 11.

**Usage**

\[
\text{Lucas}(n, \text{gmp} = \text{TRUE})
\]
Arguments

- \( n \): the number of first \( n \) entries from the sequence.
- \( gmp \): a logical; TRUE to use large number representation, FALSE otherwise.

Value

a vector of length \( n \) containing first entries from the sequence.

See Also

Fibonacci

Examples

```r
## generate first 30 Lucas numbers
print(Lucas(30))
```

Under OEIS A001006, a Motzkin number for a given \( n \) is the number of ways for drawing non-intersecting chords among \( n \) points on a circle, where the first 7 entries are 1, 1, 2, 4, 9, 21, 51.

Usage

```
Motzkin(n, gmp = TRUE)
```

Arguments

- \( n \): the number of first \( n \) entries from the sequence.
- \( gmp \): a logical; TRUE to use large number representation, FALSE otherwise.

Value

a vector of length \( n \) containing first entries from the sequence.

Examples

```r
## generate first 30 Motzkin numbers
print(Motzkin(30))
```
**Odious**

**Odious numbers**

**Description**

Under OEIS A000069, an *Odious* number has an odd number of 1’s in its binary expansion. First 6 Odious numbers are 1, 2, 4, 7, 8, 11.

**Usage**

Odious(n, gmp = TRUE)

**Arguments**

- **n**
  - the number of first \( n \) entries from the sequence.
- **gmp**
  - a logical; TRUE to use large number representation, FALSE otherwise.

**Value**

a vector of length \( n \) containing first entries from the sequence.

**See Also**

Evil

**Examples**

```r
## generate first 20 Odious numbers
print(Odious(20))
```

---

**Padovan**

**Padovan numbers**

**Description**

Under OEIS A000931, the \( n \)th Padovan number is given as

\[
F_n = F_{n-2} + F_{n-3}
\]

where the first 6 entries are 1, 0, 0, 1, 0, 1.

**Usage**

Padovan(n, gmp = TRUE)
Arguments

- **n**: the number of first \( n \) entries from the sequence.
- **gmp**: a logical; TRUE to use large number representation, FALSE otherwise.

Value

- a vector of length \( n \) containing first entries from the sequence.

Examples

```r
## generate first 30 Padovan numbers
print(Padovan(30))
```

Description

Under OEIS A002113, a **Palindrome** number is a number that remains the same when its digits are reversed. First 6 Palindrome numbers in decimal are 0, 1, 2, 3, 4, 5. This function supports various base by specifying the parameter `base` but returns are still represented in decimal.

Usage

```r
Palindromic(n, base = 10, gmp = TRUE)
```

Arguments

- **n**: the number of first \( n \) entries from the sequence.
- **base**: choice of base.
- **gmp**: a logical; TRUE to use large number representation, FALSE otherwise.

Value

- a vector of length \( n \) containing first entries from the sequence.

Examples

```r
## generate first 30 palindromic number in decimal
print(Palindromic(30))
```
Palindromic.Squares

**Description**

Under OEIS A002779, a *Palindromic square* is a number that is both Palindromic and Square. First 6 such numbers are 0, 1, 4, 9, 121, 484. It uses only the base 10 decimals.

**Usage**

Palindromic.Squares(n, gmp = TRUE)

**Arguments**

n  
the number of first n entries from the sequence.

gmp  
a logical; TRUE to use large number representation, FALSE otherwise.

**Value**

a vector of length n containing first entries from the sequence.

**Examples**

```r
# generate first 10 palindromic squares
print(Palindromic.Squares(10))
```

---

Perfect

**Description**

Under OEIS A000396, a *Perfect number* is a number whose proper divisors sum up to the extent equal to the number itself. First 6 abundant numbers are 6, 28, 496, 8128, 33550336, 8589869056.

**Usage**

Perfect(n, gmp = TRUE)

**Arguments**

n  
the number of first n entries from the sequence.

gmp  
a logical; TRUE to use large number representation, FALSE otherwise.
Value
a vector of length \( n \) containing first entries from the sequence.

See Also
Deficient, Abundant

Examples

```r
## generate first 7 Perfect numbers
print(Perfect(10))
```

---

**Perrin**  
**Perrin numbers**

Description
Under OEIS A001608, the \( n \)th Perrin number is given as

\[
F_n = F_{n-2} + F_{n-3}
\]

where the first 6 entries are 3, 0, 2, 3, 2, 5.

Usage

```
Perrin(n, gmp = TRUE)
```

Arguments

- \( n \)  
  the number of first \( n \) entries from the sequence.
- \( gmp \)  
  a logical; TRUE to use large number representation, FALSE otherwise.

Value

a vector of length \( n \) containing first entries from the sequence.

Examples

```
## generate first 30 Perrin numbers
print(Perrin(30))
```
Powerful

Powerful numbers

Description

Under OEIS A001694, a Powerful number is a positive integer such that for every prime $p$ dividing the number, $p^2$ also divides the number. First 6 powerful numbers are 1, 4, 8, 9, 16, 25.

Usage

Powerful(n, gmp = TRUE)

Arguments

- n: the number of first n entries from the sequence.
- gmp: a logical; TRUE to use large number representation, FALSE otherwise.

Value

A vector of length n containing first entries from the sequence.

Examples

```r
# generate first 20 Powerful numbers
print(Powerful(20))
```

Prime

Prime numbers

Description

Under OEIS A000040, a Prime number is a natural number with no positive divisors other than 1 and itself. First 6 prime numbers are 2, 3, 5, 7, 11, 13.

Usage

Prime(n, gmp = TRUE)

Arguments

- n: the number of first n entries from the sequence.
- gmp: a logical; TRUE to use large number representation, FALSE otherwise.
**Value**

a vector of length n containing first entries from the sequence.

**Examples**

```r
## generate first 30 Regular numbers
print(Prime(30))
```

**Description**

Under OEIS A051037, a Regular number - also known as 5-smooth - is a positive integer that even divide powers of 60, or equivalently, whose prime divisors are only 2, 3, and 5. First 6 Regular numbers are 1, 2, 3, 4, 5, 6.

**Usage**

`Regular(n, gmp = TRUE)`

**Arguments**

- **n** the number of first n entries from the sequence.
- **gmp** a logical; TRUE to use large number representation, FALSE otherwise.

**Value**

a vector of length n containing first entries from the sequence.

**Examples**

```r
## generate first 20 Regular numbers
print(Regular(20))
```
Square numbers

**Description**

Under OEIS A000290, a *Square* number is

\[ A_n = n^2 \]

for \( n \geq 0 \). First 6 Square numbers are 0, 1, 4, 9, 16, 25.

**Usage**

\[
\text{Square}(n, \text{gmp} = \text{TRUE})
\]

**Arguments**

- \( n \) the number of first \( n \) entries from the sequence.
- \( \text{gmp} \) a logical; TRUE to use large number representation, FALSE otherwise.

**Value**

a vector of length \( n \) containing first entries from the sequence.

**Examples**

```r
# generate first 20 Square numbers
print(Square(20))
```

Squarefree numbers

**Description**

Under OEIS A005117, a *Squarefree* number is a number that are not divisible by a square of a smaller integer greater than 1. First 6 Squarefree numbers are 1, 2, 3, 5, 6, 7.

**Usage**

\[
\text{Squarefree}(n, \text{gmp} = \text{TRUE})
\]

**Arguments**

- \( n \) the number of first \( n \) entries from the sequence.
- \( \text{gmp} \) a logical; TRUE to use large number representation, FALSE otherwise.
Value

A vector of length n containing first entries from the sequence.

Examples

```r
## generate first 30 Squarefree numbers
print(Squarefree(30))
```

---

Description

Under OEIS A000085, a Telephone number - also known as Involution number - is counting the number of connection patterns in a telephone system with n subscribers, or in a more mathematical term, the number of self-inverse permutations on n letters. First 6 Telephone numbers are 1, 1, 2, 4, 10, 26,

Usage

```
Telephone(n, gmp = TRUE)
```

Arguments

- `n` the number of first n entries from the sequence.
- `gmp` a logical; TRUE to use large number representation, FALSE otherwise.

Value

A vector of length n containing first entries from the sequence.

Examples

```r
## generate first 20 Regular numbers
print(Telephone(20))
```
**Thabit**

**Thabit numbers**

**Description**
Under OEIS A055010, the $n$th Thabit number is given as

$$A_n = 3 \times 2^n - 1$$

where the first 6 entries are 0, 2, 5, 11, 23, 47 with $A_0 = 0$.

**Usage**

```r
Thabit(n, gmp = TRUE)
```

**Arguments**

- **n**
  the number of first $n$ entries from the sequence.

- **gmp**
  a logical; TRUE to use large number representation, FALSE otherwise.

**Value**

a vector of length $n$ containing first entries from the sequence.

**Examples**

```r
## generate first 30 Thabit numbers
print(Thabit(30))
```

---

**Triangular**

**Triangular numbers**

**Description**
Under OEIS A000217, a Triangular number counts objects arranged in an equilateral triangle. First 6 Triangular numbers are 0, 1, 3, 6, 10, 15.

**Usage**

```r
Triangular(n, gmp = TRUE)
```

**Arguments**

- **n**
  the number of first $n$ entries from the sequence.

- **gmp**
  a logical; TRUE to use large number representation, FALSE otherwise.
Value

a vector of length \( n \) containing first entries from the sequence.

Examples

```r
## generate first 20 Triangular numbers
print(Triangular(20))
```

---

**Unusual**  
**Unusual numbers**

Description

Under OEIS A064052, an *Unusual* number is a natural number whose largest prime factor is strictly greater than square root of the number. First 6 Unusual numbers are 2, 3, 5, 6, 7, 10.

Usage

```r
Unusual(n, gmp = TRUE)
```

Arguments

- \( n \): the number of first \( n \) entries from the sequence.
- \( gmp \): a logical; TRUE to use large number representation, FALSE otherwise.

Value

a vector of length \( n \) containing first entries from the sequence.

Examples

```r
## generate first 20 Unusual numbers
print(Unusual(20))
```
Index

A000032 (Lucas), 15
A000040 (Prime), 21
A000045 (Fibonacci), 11
A000069 (Odious), 17
A000085 (Telephone), 24
A000108 (Catalan), 5
A000110 (Bell), 4
A000142 (Factorial), 9
A000165 (Factorial.Double), 11
A000217 (Triangular), 25
A000290 (Square), 23
A000396 (Perfect), 19
A000931 (Padovan), 17
A01006 (Motzkin), 16
A01147 (Factorial.Double), 11
A01608 (Perrin), 20
A01694 (Powerful), 21
A01969 (Evil), 8
A02113 (Palindromic), 18
A02779 (Palindromic.Squares), 19
A02808 (Composite), 6
A02997 (Carmichael), 5
A05100 (Deficient), 7
A05101 (Abundant), 3
A05117 (Squarefree), 23
A05165 (Factorial.Alternating), 10
A07230 (Juggler.Nsteps), 15
A07730 (Happy), 13
A046758 (Equidigital), 7
A046759 (Frugal), 12
A046760 (Extravagant), 9
A051037 (Regular), 22
A052486 (Achilles), 3
A055100 (Thabit), 25
A064052 (Unusual), 26
A094683 (Juggler), 13
A094716 (Juggler.Largest), 14
Abundant, 3, 7, 20
Achilles, 3

Bell, 4
Carmichael, 5
Catalan, 5
Composite, 6
Deficient, 3, 7, 20
Economical (Frugal), 12
Equidigital, 7, 9, 12
Evil, 8, 17
Extravagant, 8, 9, 12
Factorial, 9, 10, 11
Factorial.Alternating, 10
Factorial.Double, 11
Fibonacci, 11, 16
Frugal, 8, 9, 12
Happy, 13
Involution (Telephone), 24
Juggler, 13, 14, 15
Juggler.Largest, 14
Juggler.Nsteps, 15
Lucas, 15
Motzkin, 16
Odious, 8, 17
Padovan, 17
Palindromic, 18
Palindrome.Squares, 19
Perfect, 3, 7, 19
Perrin, 20
Powerful, 21
Prime, 21
Regular, 22
Segner (Catalan), 5
Square, 23
Squarefree, 23

Telephone, 24
Thabit, 25
Triangular, 25

Unusual, 26

Wasteful (Extravagant), 9

Zseq-package, 2