1 Integer partitions

```
Preliminary remark:  This wxMaxima worksheet is part of the project
        Sommerschule "Lust auf Mathematik"
        Humboldt Universität, Berlin
        June 2024
The project is presented in the following files:
LaTeX-generated file on Fibonacci numbers, powerseries and generating functions:
  https://jochen-ziegenbalg.github.io/materialien/Manuskripte/Fib-Pot-erzFkt.pdf
LaTeX-generated file on integer partitions:
  https://jochen-ziegenbalg.github.io/materialien/Manuskripte/Integer-Partitions.pdf
Maxima-generated file on integer partitions:
 https://jochen-ziegenbalg.github.io/materialien/Manuskripte/Partitions-CAS-Maxima.pdf
```

1.1 Changing money and stamping letters - according to an idea of Leonhard Euler

In this worksheet we demonstrate an ingenious idea of Leonhard Euler.
(Wikipedia: Leonhard Euler, 1707-1783, was a Swiss mathematician, physicist, astronomer, geographer, logician, and engineer who founded the studies of graph theory and topology and made pioneering and influential discoveries in many other branches of mathematics such as analytic number theory, complex analysis, and infinitesimal calculus.)

The stamping problem:

Let us imagine we have to stamp a letter. Let us assume that the postage is 85 Ct.
We have the following stock of stamps: 6 stamps of 1 Ct, 5 stamps of 5 Ct, 3 stamps of 10 Ct, 2 stamps of 20 Ct 1 stamp of 50 Ct.
In how many different ways can the letter be stamped - if no distinction is made between stamps of the same value (and if it does not matter where on the letter the stamps are placed).

With a little (systematic) trial and error, we arrive at the following 10 possibilities:

```
\label{stamping-85}
  1:   50+20 + 10 + 5
  2:   50 + 20 + 10 + 1 + 1 + 1 + 1 + 1
  3:   50 + 20  + 5 + 5 + 5
  4:   50 + 20 + 5 + 5 + 1 + 1 + 1 + 1 + 1
  5:   50 + 10 + 10 + 10 + 5
  6:   50 + 10 + 10 + 10 + 1 + 1 + 1 + 1 + 1
  7:   50 + 10 + 10 + 5 + 5 + 5
  8:   50 + 10 + 10 + 5 + 5 + 1 + 1 + 1 + 1 + 1
  9:   50 + 10 + 5 + 5 + 5 + 5
 10:   50 + 10 + 5 + 5 + 5 + 5 + 1 + 1 + 1 + 1 + 1
 11:   20 + 20 + 10 + 10 + 10 + 5 + 5 + 5
 12:   20 + 20 + 10 + 10 + 10 + 5 + 5 + 1 + 1 + 1 + 1 + 1
 13:   20 + 20 + 10 + 10 + 5 + 5 + 5 + 5 + 5
 14:   20 + 20 + 10 + 10 + 5 + 5 + 5 + 5 + 1 + 1 + 1 + 1 + 1
```

```
\label{stamping-45}
20 + 20 + 5
20 + 20 + 1 + 1 + 1 + 1 + 1
20 + 10 + 10 + 5
20 + 10 + 10 + 1 + 1 + 1 + 1 + 1
20 + 10 +  5 + 5 + 5
20 + 10 +  5 + 5 + 1 + 1 + 1 + 1 + 1
20 +  5 +  5 + 5 + 5 + 1 + 1 + 1 + 1 + 1
10 + 10 + 10 + 5 + 5 + 5
10 + 10 + 10 + 5 + 5 + 1 + 1 + 1 + 1 + 1
10 + 10 +  5 + 5 +  5 + 5 +1 + 1 + 1 + 1 + 1
```

```
powerdisp:true $      /* Maxima option:  display powers of X in
                         ascending order in this worksheet */
```

In order to systematically find all the possible ways of stamping the letter, we first consider the following product of powers of X.

(1+X+X^2+X^3+X^4+X^5) * (1+X^5+X^10)

The function "expand" computes the product by applying the distributive law. Thus, to compute the product we have to sum up all the terms X^a * X^b, where X^a is taken from the first factor and X^b is taken from the second factor.

```
expand( (1 + X + X^2 + X^3 + X^4 + X^5) * (1 + X^5 + X^10) );
```

$$1+X+X^2+X^3+X^4+2*X^5+X^6+X^7+X^8+X^9+2*X^{10}+X^{11}+X^{12}+X^{13}+X^{14}+X^{15}$$

Note: The coefficient 2 in front of X^10 results from the following summands (or parts): X^2*X^5 and 1*X^10.
(Sometimes, to make things clearer, it is helpful to think of 1 as of X^0 or even 1*X^0.)

```
Let's get back to stamping the 85-Ct postage letter.

We consider the product of polynomials:

(1+X+X^2+X^3+X^4+X^5+X^6) * (1+X^5+(X^5)^2+(X^5)^3 +(X^5)^4 +(X^5)^5)    *
(1+X^10+(X^10)^2+X^30) * (1+X^20+(X^20)^2) * (1+X^50)

Each of the factors relates to the stamps of a particular value in the following way:

(1+X+X^2+X^3+X^4+X^5+X^6)   <--->  6 stamps of  1 Ct each
(1+X^5+X^10+X^15+X^20+X^25) = ((1 + X^5 + (X^5)^2 + (X^5)^3 + (X^5)^4 + (X^5)^5 )
                            <--->   5  stamps of value 5 Ct each
(1+X^10+X^20+X^30) = (1 + X^10 + (X^10)^2 + (X^10)^3 )
                            <--->   3 stamps of value 10 Ct each
(1+X^20+X^40) = (1 + X^20 + (X^20)^2 )
                            <--->  2 stamps of value 20 Ct each
(1+X^50)                    <--->  2 stamps of value 50 Ct each
```

The following call shows all the possibilities of stamping with the given stock.

```
Cstock :
expand( (1 + X + X^2  + X^3 + X^4 + X^5 + X^6) *
   (1 + X^5 + (X^5)^2 + (X^5)^3 + (X^5)^4 + (X^5)^5 )   *
     (1 + X^10 + (X^10)^2 + X^(3*10) ) *
        (1 + X^20 + (X^20)^2 ) *
          (1 + X^50) ) ;

/*  Before you look at the evaluation, think about the following
    Problem:  What is the highest power of X in this term?  */
```

$$1+X+X^{2}+X^{3}+X^{4}+2*X^{5}+2*X^{6}+X^{7}+X^{8}+X^{9}+3*X^{10}+3*X^{11}+2*X^{12}+2*X^{13}+2*X^{14}+4*X^{15}+4*X^{16}+2*X^{17}+2*X^{18}+2*X^{19}+6*X^{20}+6*X^{21}+4*X^{22}+4*X^{23}+4*X^{24}+8*X^{25}+8*X^{26}+4*X^{27}+4*X^{28}+4*X^{29}+9*X^{30}+9*X^{31}+5*X^{32}+5*X^{33}+5*X^{34}+10*X^{35}+10*X^{36}+5*X^{37}+5*X^{38}+5*X^{39}+11*X^{40}+11*X^{41}+6*X^{42}+6*X^{43}+6*X^{44}+12*X^{45}+12*X^{46}+6*X^{47}+6*X^{48}+6*X^{49}+13*X^{50}+13*X^{51}+7*X^{52}+7*X^{53}+7*X^{54}+14*X^{55}+14*X^{56}+7*X^{57}+7*X^{58}+7*X^{59}+14*X^{60}+14*X^{61}+7*X^{62}+7*X^{63}+7*X^{64}+14*X^{65}+14*X^{66}+7*X^{67}+7*X^{68}+7*X^{69}+15*X^{70}+15*X^{71}+8*X^{72}+8*X^{73}+8*X^{74}+16*X^{75}+16*X^{76}+8*X^{77}+8*X^{78}+8*X^{79}+15*X^{80}+15*X^{81}+7*X^{82}+7*X^{83}+7*X^{84}+14*X^{85}+14*X^{86}+7*X^{87}+7*X^{88}+7*X^{89}+14*X^{90}+14*X^{91}+7*X^{92}+7*X^{93}+7*X^{94}+14*X^{95}+14*X^{96}+7*X^{97}+7*X^{98}+7*X^{99}+13*X^{100}+13*X^{101}+6*X^{102}+6*X^{103}+6*X^{104}+12*X^{105}+12*X^{106}+6*X^{107}+6*X^{108}+6*X^{109}+11*X^{110}+11*X^{111}+5*X^{112}+5*X^{113}+5*X^{114}+10*X^{115}+10*X^{116}+5*X^{117}+5*X^{118}+5*X^{119}+9*X^{120}+9*X^{121}+4*X^{122}+4*X^{123}+4*X^{124}+8*X^{125}+8*X^{126}+4*X^{127}+4*X^{128}+4*X^{129}+6*X^{130}+6*X^{131}+2*X^{132}+2*X^{133}+2*X^{134}+4*X^{135}+4*X^{136}+2*X^{137}+2*X^{138}+2*X^{139}+3*X^{140}+3*X^{141}+X^{142}+X^{143}+X^{144}+2*X^{145}+2*X^{146}+X^{147}+X^{148}+X^{149}+X^{150}+X^{151}$$

In Maxima, the function for extracting the coefficient of a polynomial is "coeff"; for example:

```
coeff(%, X^85);                  /*  The symbol % returns the last output */
```

14

```
coeff(Cstock, X^85);
```

14

Clearly, all letters with postage 1 Ct to postage 151 Ct can be stamped. Some of them by using different stamps.
The term with X^85 relates to the postage of 85 Ct. Its coefficient 14 shows that there are 14 ways of picking 5 Terms,from each of the following parentheses.

Observe: the coefficient of X^85 is in accordance with the listing (above) in \label{stamping-85}.

1.1.1 Modification: Having a stock of distinct stamps: 1 stamp of each kind

Modification of the problem: What if the postage were 8 Ct and if there were exactly one stamp each of 1 Ct, 2 Ct, 3 Ct, 4 Ct, 5 Ct, 6 Ct, 7 Ct and 8 Ct available?

In this case, the analog of Euler's expression would be:

```
expand( (1+X)*(1+X^2)*(1+X^3)*(1+X^4)*(1+X^5)*(1+X^6)*(1+X^7)*(1+X^8) );
```

$$1+X+X^{2}+2*X^{3}+2*X^{4}+3*X^{5}+4*X^{6}+5*X^{7}+6*X^{8}+7*X^{9}+8*X^{10}+9*X^{11}+10*X^{12}+11*X^{13}+12*X^{14}+13*X^{15}+13*X^{16}+13*X^{17}+14*X^{18}+13*X^{19}+13*X^{20}+13*X^{21}+12*X^{22}+11*X^{23}+10*X^{24}+9*X^{25}+8*X^{26}+7*X^{27}+6*X^{28}+5*X^{29}+4*X^{30}+3*X^{31}+2*X^{32}+2*X^{33}+X^{34}+X^{35}+X^{36}$$

The coefficient of X^8 shows that there are 6 ways of stamping the postage of 8 Ct (using only 1 stamp of each value):   8, 7+1, 6+2, 5+3, 5+2+1, 4+3+1

1.1.2 Two stamps of each kind

If there are 2 stamps of each in the stock the expansion of Euler's polynomial shows that there are 13 possibilities:

```
expand(
        ( 1+X^1 + (X^1)^2 ) *
        ( 1+X^2 + (X^2)^2 ) *
        ( 1+X^3 + (X^3)^2 ) *
        ( 1+X^4 + (X^4)^2 ) *
        ( 1+X^5 + (X^5)^2 ) *
        ( 1+X^6 + (X^6)^2 ) *
        ( 1+X^7 + (X^7)^2 ) *
        ( 1+X^8 + (X^8)^2 ) ) ;
```

$$1+X+2*X^{2}+2*X^{3}+4*X^{4}+5*X^{5}+7*X^{6}+9*X^{7}+13*X^{8}+15*X^{9}+20*X^{10}+23*X^{11}+30*X^{12}+34*X^{13}+42*X^{14}+48*X^{15}+58*X^{16}+64*X^{17}+75*X^{18}+82*X^{19}+95*X^{20}+103*X^{21}+116*X^{22}+124*X^{23}+138*X^{24}+145*X^{25}+158*X^{26}+165*X^{27}+$$

$$178X^{28}+183X^{29}+194X^{30}+197X^{31}+207X^{32}+207X^{33}+214X^{34}+213X^{35}+217X^{36}+213X^{37}+214X^{38}+207X^{39}+207X^{40}+197X^{41}+194X^{42}+183X^{43}+178X^{44}+165X^{45}+158X^{46}+145X^{47}+138X^{48}+124X^{49}+116X^{50}+103X^{51}+95X^{52}+82X^{53}+75X^{54}+64X^{55}+58X^{56}+48X^{57}+42X^{58}+34X^{59}+30X^{60}+23X^{61}+20X^{62}+15X^{63}+13X^{64}+9X^{65}+7X^{66}+5X^{67}+4X^{68}+2X^{69}+2X^{70}+X^{71}+X^{72}$$

```
coeff(%, X^8);
```

13

1.1.3 Three stamps of each kind

```
expand(
        ( 1+X^1 + (X^1)^2 + (X^1)^3  ) *
        ( 1+X^2 + (X^2)^2 + (X^2)^3  ) *
        ( 1+X^3 + (X^3)^2 + (X^3)^3  ) *
        ( 1+X^4 + (X^4)^2 + (X^4)^3  ) *
        ( 1+X^5 + (X^5)^2 + (X^5)^3  ) *
        ( 1+X^6 + (X^6)^2 + (X^6)^3  ) *
        ( 1+X^7 + (X^7)^2 + (X^7)^3  ) *
        ( 1+X^8 + (X^8)^2 + (X^8)^3  ) ) ;
```

$$1+X+2X^{2}+3X^{3}+4X^{4}+6X^{5}+9X^{6}+12X^{7}+16X^{8}+21X^{9}+27X^{10}+34X^{11}+43X^{12}+53X^{13}+65X^{14}+79X^{15}+94X^{16}+112X^{17}+132X^{18}+154X^{19}+179X^{20}+207X^{21}+236X^{22}+269X^{23}+304X^{24}+342X^{25}+382X^{26}+426X^{27}+471X^{28}+519X^{29}+569X^{30}+621X^{31}+674X^{32}+730X^{33}+785X^{34}+843X^{35}+901X^{36}+958X^{37}+1015X^{38}+1072X^{39}+1127X^{40}+1181X^{41}+1234X^{42}+1283X^{43}+1330X^{44}+1374X^{45}+1414X^{46}+1450X^{47}+1483X^{48}+1510X^{49}+1534X^{50}+1552X^{51}+1565X^{52}+1572X^{53}+1576X^{54}+1572X^{55}+1565X^{56}+1552X^{57}+1534X^{58}+1510X^{59}+1483X^{60}+1450X^{61}+1414X^{62}+1374X^{63}+1330X^{64}+1283X^{65}+1234X^{66}+1181X^{67}+1127X^{68}+1072X^{69}+1015X^{70}+958X^{71}+901X^{72}+843X^{73}+785X^{74}+730X^{75}+674X^{76}+621X^{77}+569X^{78}+519X^{79}+471X^{80}+426X^{81}+382X^{82}+342X^{83}+304X^{84}+269X^{85}+236X^{86}+207X^{87}+179X^{88}+154X^{89}+132X^{90}+112X^{91}+94X^{92}+79X^{93}+65X^{94}+53X^{95}+43X^{96}+34X^{97}+27X^{98}+21X^{99}+16X^{100}+12X^{101}+9X^{102}+6X^{103}+4X^{104}+3X^{105}+2X^{106}+X^{107}+X^{108}$$

```
coeff(%, X^8);
```

16

Exercises:
(1.) List all the 13 (respectively 16) possibilitiesfrom above.
(2.) Do the same thing for 4 stamps of each kind.

```
/* using the summation and product functions of Maxima */

sum((X^j)^k, k, 0, 5);
```

$$1+X^{j}+X^{2j}+X^{3j}+X^{4j}+X^{5j}$$

```
prod(sum((X^j)^k, k, 0, 2), j, 1, 8);                 /* max-equal  = 2 */
```

$$(1+X+X^{2})(1+X^{2}+X^{4})(1+X^{3}+X^{6})(1+X^{4}+X^{8})(1+X^{5}+X^{10})(1+X^{6}+X^{12})(1+X^{7}+X^{14})(1+X^{8}+X^{16})$$

```
expand(%);
```

$$1+X+2X^{2}+2X^{3}+4X^{4}+5X^{5}+7X^{6}+9X^{7}+13X^{8}+15X^{9}+20X^{10}+23X^{11}+30X^{12}+34X^{13}+42X^{14}+48X^{15}+58X^{16}+64X^{17}+75X^{18}+82X^{19}+95X^{20}+103X^{21}+116X^{22}+124X^{23}+138X^{24}+145X^{25}+158X^{26}+165X^{27}+178X^{28}+183X^{29}+194X^{30}+197X^{31}+207X^{32}+207X^{33}+214X^{34}+213X^{35}+217X^{36}+213X^{37}+214X^{38}+207X^{39}+207X^{40}+197X^{41}+194X^{42}+183X^{43}+178X^{44}+165X^{45}+158X^{46}+145X^{47}+138X^{48}+124X^{49}+116X^{50}+103X^{51}+95X^{52}+82X^{53}+75X^{54}+64X^{55}+58X^{56}+48X^{57}+42X^{58}+34X^{59}+30X^{60}+23X^{61}+20X^{62}+15X^{63}+13X^{64}+9X^{65}+7X^{66}+5X^{67}+4X^{68}+2X^{69}+2X^{70}+X^{71}+X^{72}$$

```
coeff(%, X^8);
```

13

```
product(sum((X^j)^k, j, 0, 3), k, 1, 8 ) ;           /* max-equal  = 3 */
```

$$(1+X+X^{2}+X^{3})(1+X^{2}+X^{4}+X^{6})(1+X^{3}+X^{6}+X^{9})(1+X^{4}+X^{8}+X^{12})(1+X^{5}+X^{10}+X^{15})(1+X^{6}+X^{12}+X^{18})(1+X^{7}+X^{14}+X^{21})(1+X^{8}+X^{16}+X^{24})$$

```
expand(%);
```

$$1+X+2X^{2}+3X^{3}+4X^{4}+6X^{5}+9X^{6}+12X^{7}+16X^{8}+21X^{9}+27X^{10}+34X^{11}+43X^{12}+53X^{13}+65X^{14}+79X^{15}+94X^{16}+112X^{17}+132X^{18}+154X^{19}+179X^{20}+207X^{21}+236X^{22}+269X^{23}+304X^{24}+342X^{25}+382X^{26}+426X^{27}+471X^{28}+519X^{29}+569X^{30}+621X^{31}+674X^{32}+730X^{33}+785X^{34}+843X^{35}+901X^{36}+958X^{37}+1015X^{38}+1072X^{39}+1127X^{40}+1181X^{41}+1234X^{42}+1283X^{43}+1330X^{44}+1374X^{45}+1414X^{46}+1450X^{47}+1483X^{48}+1510X^{49}+1534X^{50}+1552X^{51}+1565X^{52}+1572X^{53}+1576X^{54}+1572X^{55}+1565X^{56}+1552X^{57}+1534X^{58}+1510X^{59}+1483X^{60}+1450X^{61}+1414X^{62}+1374X^{63}+1330X^{64}+1283X^{65}+1234X^{66}+1181X^{67}+1127X^{68}+1072X^{69}+1015X^{70}+958X^{71}+901X^{72}+843X^{73}+785X^{74}+730X^{75}+674X^{76}+621X^{77}+569X^{78}+519X^{79}+471X^{80}+426X^{81}+382X^{82}+342X^{83}+304X^{84}+269X^{85}+236X^{86}+207X^{87}+179X^{88}+154X^{89}+132X^{90}+112X^{91}+94X^{92}+79X^{93}+65X^{94}+53X^{95}+43X^{96}+34X^{97}+27X^{98}+21X^{99}+16X^{100}+12X^{101}+9X^{102}+6X^{103}+4X^{104}+3X^{105}+2X^{106}+X^{107}+X^{108}$$

```
coeff(%, X^8);
```

16

```
/* concatenation of the functions */

coeff(expand(product(sum((X^j)^k, j, 0, 4), k, 1, 8 ) ), X^8)          /* max-equal  = 4 */  ;
```

19

1.2 Modification: unlimited supply of stamps

Modification and generalization of the stamping problem: Let us assume that the postage is 45 Ct instead of 85 Ct.
(Otherwise, the numbers would get unwieldingly high in the following text.)

Now suppose we had an unlimited supply of 1 Ct, 5 Ct 10 Ct and 20 Ct stamps. Then by the same argumentation we have the following number of ways to stamp the 45 Ct postage letter. (The upper limits for "sum" are chosen so that
        the power of X times the upper limit
is the highest value less than or equal to 45.)

```
C1 : sum(X^k, k, 0, 45) * sum((X^5)^k, k, 0, 9) * sum((X^10)^k, k, 0, 4) * sum((X^20)^k, k, 0, 2) ;
```

$$(1+X^{20}+X^{40})*(1+X^{10}+X^{20}+X^{30}+X^{40})*(1+X^{5}+X^{10}+X^{15}+X^{20}+X^{25}+X^{30}+X^{35}+X^{40}+X^{45})*(1+X+X^{2}+X^{3}+X^{4}+X^{5}+X^{6}+X^{7}+X^{8}+X^{9}+X^{10}+X^{11}+X^{12}+X^{13}+X^{14}+X^{15}+X^{16}+X^{17}+X^{18}+X^{19}+X^{20}+X^{21}+X^{22}+X^{23}+X^{24}+X^{25}+X^{26}+X^{27}+X^{28}+X^{29}+X^{30}+X^{31}+X^{32}+X^{33}+X^{34}+X^{35}+X^{36}+X^{37}+X^{38}+X^{39}+X^{40}+X^{41}+X^{42}+X^{43}+X^{44}+X^{45})$$

```
expand(C1) ;
```

$$1+X+X^{2}+X^{3}+X^{4}+2*X^{5}+2*X^{6}+2*X^{7}+2*X^{8}+2*X^{9}+4*X^{10}+4*X^{11}+4*X^{12}+4*X^{13}+4*X^{14}+6*X^{15}+6*X^{16}+6*X^{17}+6*X^{18}+6*X^{19}+10*X^{20}+10*X^{21}+10*X^{22}+10*X^{23}+10*X^{24}+14*X^{25}+14*X^{26}+14*X^{27}+14*X^{28}+14*X^{29}+20*X^{30}+20*X^{31}+20*X^{32}+20*X^{33}+20*X^{34}+26*X^{35}+26*X^{36}+26*X^{37}+26*X^{38}+26*X^{39}+35*X^{40}+35*X^{41}+35*X^{42}+35*X^{43}+35*X^{44}+44*X^{45}+43*X^{46}+43*X^{47}+43*X^{48}+43*X^{49}+53*X^{50}+52*X^{51}+52*X^{52}+52*X^{53}+52*X^{54}+62*X^{55}+60*X^{56}+60*X^{57}+60*X^{58}+60*X^{59}+71*X^{60}+69*X^{61}+69*X^{62}+69*X^{63}+69*X^{64}+80*X^{65}+76*X^{66}+76*X^{67}+76*X^{68}+76*X^{69}+86*X^{70}+82*X^{71}+82*X^{72}+82*X^{73}+82*X^{74}+92*X^{75}+86*X^{76}+86*X^{77}+86*X^{78}+86*X^{79}+95*X^{80}+89*X^{81}+89*X^{82}+89*X^{83}+89*X^{84}+98*X^{85}+89*X^{86}+89*X^{87}+89*X^{88}+89*X^{89}+95*X^{90}+86*X^{91}+86*X^{92}+86*X^{93}+86*X^{94}+92*X^{95}+82*X^{96}+82*X^{97}+82*X^{98}+82*X^{99}+86*X^{100}+76*X^{101}+76*X^{102}+76*X^{103}+76*X^{104}+80*X^{105}+69*X^{106}+69*X^{107}+69*X^{108}+69*X^{109}+71*X^{110}+60*X^{111}+60*X^{112}+60*X^{113}+60*X^{114}+62*X^{115}+52*X^{116}+52*X^{117}+52*X^{118}+52*X^{119}+53*X^{120}+43*X^{121}+43*X^{122}+43*X^{123}+43*X^{124}+44*X^{125}+35*X^{126}+35*X^{127}+35*X^{128}+35*X^{129}+35*X^{130}+26*X^{131}+26*X^{132}+26*X^{133}+26*X^{134}+26*X^{135}+20*X^{136}+20*X^{137}+20*X^{138}+20*X^{139}+20*X^{140}+14*X^{141}+14*X^{142}+14*X^{143}+14*X^{144}+14*X^{145}+10*X^{146}+10*X^{147}+10*X^{148}+10*X^{149}+10*X^{150}+6*X^{151}+6*X^{152}+6*X^{153}+6*X^{154}+6*X^{155}+4*X^{156}+4*X^{157}+4*X^{158}+4*X^{159}+4*X^{160}+2*X^{161}+2*X^{162}+2*X^{163}+2*X^{164}+2*X^{165}+X^{166}+X^{167}+X^{168}+X^{169}+X^{170}$$

```
coeff(%, X^45);
```

44

The coefficient 44 of X^45 shows, that with this "large enough" supply of the stamps there are 44 ways of stamping the 45 Ct postage letter.

Raising the esponents in

```
   sum(X^k, k, 0, 45) * sum((X^5)^k, k, 0, 9) * sum((X^10)^k, k, 0, 4)*
     sum((X^20)^k, k, 0, 2);
```

for instance like this

```
   sum(X^k, k, 0, 50) * sum((X^5)^k, k, 0, 50) * sum((X^10)^k, k, 0, 50)*
     sum((X^20)^k, k, 0, 50);
```

will not give new stamping solutions, because the new exponents will be too high and summing them up will lead to (for the problem irrelevant) sums beyond 45.
(The highest exponent would in this case be 1800 (= 50*1 + 50*5 + 50*10 + 50*20)).

```
C2 : expand(sum(X^k, k, 0, 50) * sum((X^5)^k, k, 0, 50) * sum((X^10)^k, k, 0, 50)*
    sum((X^20)^k, k, 0, 50) ) $

/* The result is a very long expression of 1801 summands.
   The $ - character is for "no display"  */  ;
```

```
length(C2);
```

1801

```
coeff(C2, X^45);     /*  test for confirmation   */
```

44

```
coeff(C2, X, 45);                  /* call by alternate syntax */
```

44

Similarly, by raising the exponents up to infinity, we get no new solution for our stamping problem.

What we get is the (formal) expression ("inf" being Maxima's symbol for "infinity"):

```
C3 : sum(X^k, k, 0, inf) * sum((X^5)^k, k, 0, inf) * sum((X^10)^k, k, 0, inf)*
sum((X^20)^k, k, 0, inf)  ;
```

$$\left(\sum_{k=0}^{\infty}(X^{k})\right)*\left(\sum_{k=0}^{\infty}(X^{5*k})\right)*\left(\sum_{k=0}^{\infty}(X^{10*k})\right)*\sum_{k=0}^{\infty}(X^{20*k})$$

```
/*    Reserve    C4 : sum(X^k, k, 0, inf) * sum((X^5)^k, k, 0, inf) *
sum((X^10)^k, k, 0, inf)* sum((X^20)^k, k, 0, inf)*sum((X^50)^k, k, 0, inf) */;
```

Does this help?
Yes! If we take into account the following fact on power series:
        sum(Y^k, k, 0, inf) = 1 / (1-Y).
So, P6 can be written as:

```
T1 : (1/(1-X))*(1/(1-X^5))*(1/(1-X^10))*(1/(1-X^20)) ;
```

$$\frac{1}{(1-X)*(1-X^{5})*(1-X^{10})*(1-X^{20})}$$

1.3 Using Taylor series

For a brief excursion on Taylor series - see the manuscript on Fibonacci Numbers, Power Series, Generating Funktions:
Taylor series are implemented in Maxima; syntax: taylor(expr, x, a, n); see the following example.

Taking the Taylor series of T1 confirms the 44 ways of stamping (check the coefficient of X^45):

```
T2 : taylor(T1, X, 0, 50);
```

$$1+X+X^{2}+X^{3}+X^{4}+2*X^{5}+2*X^{6}+2*X^{7}+2*X^{8}+2*X^{9}+4*X^{10}+4*X^{11}+4*X^{12}+4*X^{13}+4*X^{14}+6*X^{15}+6*X^{16}+6*X^{17}+6*X^{18}+6*X^{19}+10*X^{20}+10*X^{21}+10*X^{22}+10*X^{23}+10*X^{24}+14*X^{25}+14*X^{26}+14*X^{27}+14*X^{28}+14*X^{29}+20*X^{30}+20*X^{31}+20*X^{32}+20*X^{33}+20*X^{34}+26*X^{35}+26*X^{36}+26*X^{37}+26*X^{38}+26*X^{39}+35*X^{40}+35*X^{41}+35*X^{42}+35*X^{43}+35*X^{44}+44*X^{45}+44*X^{46}+44*X^{47}+44*X^{48}+44*X^{49}+56*X^{50}+...$$

```
coeff(T2, X^45);
```

44

So, the Taylor method confirms that there are 44 ways of stamping the letter (with an unlimited supply of 1 and 5 an 10 and 20 - Ct stamps).

Check: Using the same method in the case of "postage = 85 Ct" we have:

```
T3 : taylor( (1/(1-X))*(1/(1-X^5))*(1/(1-X^10))*(1/(1-X^20))*(1/(1-X^50)), X, 0, 100) ;
```

$$1+X+X^{2}+X^{3}+X^{4}+2*X^{5}+2*X^{6}+2*X^{7}+2*X^{8}+2*X^{9}+4*X^{10}+4*X^{11}+4*X^{12}+4*X^{13}+4*X^{14}+6*X^{15}+6*X^{16}+6*X^{17}+6*X^{18}+6*X^{19}+10*X^{20}+10*X^{21}+10*X^{22}+10*X^{23}+10*X^{24}+14*X^{25}+14*X^{26}+14*X^{27}+14*X^{28}+14*X^{29}+20*X^{30}+20*X^{31}+20*X^{32}+20*X^{33}+20*X^{34}+26*X^{35}+26*X^{36}+26*X^{37}+26*X^{38}+26*X^{39}+35*X^{40}+35*X^{41}+35*X^{42}+35*X^{43}+35*X^{44}+44*X^{45}+44*X^{46}+44*X^{47}+44*X^{48}+44*X^{49}+57*X^{50}+57*X^{51}+57*X^{52}+57*X^{53}+57*X^{54}+70*X^{55}+70*X^{56}+70*X^{57}+70*X^{58}+70*X^{59}+88*X^{60}+88*X^{61}+88*X^{62}+88*X^{63}+88*X^{64}+106*X^{65}+106*X^{66}+106*X^{67}+106*X^{68}+106*X^{69}+130*X^{70}+130*X^{71}+130*X^{72}+130*X^{73}+130*X^{74}+154*X^{75}+154*X^{76}+154*X^{77}+154*X^{78}+154*X^{79}+185*X^{80}+185*X^{81}+185*X^{82}+185*X^{83}+185*X^{84}+216*X^{85}+216*X^{86}+216*X^{87}+216*X^{88}+216*X^{89}+255*X^{90}+255*X^{91}+255*X^{92}+255*X^{93}+255*X^{94}+294*X^{95}+294*X^{96}+294*X^{97}+294*X^{98}+294*X^{99}+343*X^{100}+...$$

```
coeff(%, X, 85);
```

216

If there were an unlimited stock of all stamps of 1 Ct, 2 Ct, 3 Ct, ... 85 Ct we would, for example, get

```
coeff(expand(product(sum((X^k)^j, k, 0, 15), j, 1, 15) ), X^15 ) ;
```

176

```
PA[15];
```

PA_{15}

1.4 (First) Summary

By applying the "Euler method" we get

```
E1(n) := product(sum((X^k)^j, k, 0, n), j, 1, n);
```

$$E1(n):=\prod_{j=1}^{n}\left(\sum_{k=0}^{n}\left((X^{k})^{j}\right)\right)$$

```
E2(n) := product(sum((X^k)^j, k, 0, inf), j, 1, n);
```

$$E2(n):=\prod_{j=1}^{n}\left(\sum_{k=0}^{\infty}\left((X^{k})^{j}\right)\right)$$

```
E3(n) := product(1/(1-X^j), j, 1, n);
```

$$E3(n):=\prod_{j=1}^{n}\left(\frac{1}{1-X^{j}}\right)$$

Some concrete cases:

```
E1(10);
```

$$(1+X+X^{2}+X^{3}+X^{4}+X^{5}+X^{6}+X^{7}+X^{8}+X^{9}+X^{10})*(1+X^{2}+X^{4}+X^{6}+X^{8}+X^{10}+X^{12}+X^{14}+X^{16}+X^{18}+X^{20})*(1+X^{3}+X^{6}+X^{9}+X^{12}+X^{15}+X^{18}+X^{21}+X^{24}+X^{27}+X^{30})*(1+X^{4}+X^{8}+X^{12}+X^{16}+X^{20}+X^{24}+X^{28}+X^{32}+X^{36}+X^{40})*(1+X^{5}+X^{10}+X^{15}+X^{20}+X^{25}+X^{30}+X^{35}+X^{40}+X^{45}+X^{50})*(1+X^{6}+X^{12}+X^{18}+X^{24}+X^{30}+X^{36}+X^{42}+X^{48}+X^{54}+X^{60})*(1+X^{7}+X^{14}+X^{21}+X^{28}+X^{35}+X^{42}+X^{49}+X^{56}+X^{63}+X^{70})*(1+X^{8}+X^{16}+X^{24}+X^{32}+X^{40}+X^{48}+X^{56}+X^{64}+X^{72}+X^{80})*(1+X^{9}+X^{18}+X^{27}+X^{36}+X^{45}+X^{54}+X^{63}+X^{72}+X^{81}+X^{90})*(1+X^{10}+X^{20}+X^{30}+X^{40}+X^{50}+X^{60}+X^{70}+X^{80}+X^{90}+X^{100})$$

```
E2(10);
```

$$\left(\sum_{k=0}^{\infty}(X^{k})\right)*\left(\sum_{k=0}^{\infty}(X^{2*k})\right)*\left(\sum_{k=0}^{\infty}(X^{3*k})\right)*\left(\sum_{k=0}^{\infty}(X^{4*k})\right)*\left(\sum_{k=0}^{\infty}(X^{5*k})\right)*\left(\sum_{k=0}^{\infty}(X^{6*k})\right)*\left(\sum_{k=0}^{\infty}(X^{7*k})\right)*\left(\sum_{k=0}^{\infty}(X^{8*k})\right)*\left(\sum_{k=0}^{\infty}(X^{9*k})\right)*\sum_{k=0}^{\infty}(X^{10*k})$$

```
E3(10);
```

$$\frac{1}{(1-X)*(1-X^2)*(1-X^3)*(1-X^4)*(1-X^5)*(1-X^6)*(1-X^7)*(1-X^8)*(1-X^9)*(1-X^{10})}$$

Coefficients by Taylor expansion:

```
PT1(n) := coeff(expand(taylor(product(sum((X^k)^j, k, 0, n), j, 1, n) ) ), X^n ) ;
```

$$PT1(n):=coeff\left(expand\left(taylor\left(\prod_{j=1}^{n}\left(\sum_{k=0}^{n}\left((X^k)^j\right)\right)\right)\right),X^n\right)$$

```
PT2(n) := coeff(expand(taylor(product(sum((X^k)^j, k, 0, inf), j, 1, n) ) ), X^n ) ;
```

$$PT2(n):=coeff\left(expand\left(taylor\left(\prod_{j=1}^{n}\left(\sum_{k=0}^{\infty}\left((X^k)^j\right)\right)\right)\right),X^n\right)$$

```
PT3(n) := coeff(expand(taylor(product(1/(1-X^j), j, 1, n), X, 0, n) ), X^n );
```

$$PT3(n):=coeff\left(expand\left(taylor\left(\prod_{j=1}^{n}\left(\frac{1}{1-X^j}\right),X,0,n\right)\right),X^n\right)$$

```
PT1(20);
    627
PT2(20);            /* no evaluation possible in this form */
    0
PT3(20);
    627
PA[20];             /*  PA is a function defined below  */
```

PA_{20}

2 Integer partitions

In section 1, we have found a way of splitting up the number 85 into a sum of the integers 1, 5, 10, 20 and 50.
Next we will work on a method of splitting up any integer into any sum of any kind of integers.

An integer partition of an integer n is a collection of integers with their sum being n.

For instance, 3+1, 2+2, 2+1+1, 1+1+1+1 are integer partitions of the number 4. But also, the "sum" 4 with only one summand belongs to this partition. So, there are 5 integer partitions of 4.

This section is about generating all inter partitions of a given integer.

In the following, we will discuss
- the integer partitions per se, and
- the number of the integer partitions.

All integer partitions will be implemented as a list of lists, and the numbers of their elements (also called length) are just, well, numbers.

The list of integer partitions will be denoted by PL (partitions-list), PL_full and PL_std (description see below). These are functions each one resulting in a list of lists containing the partitions in different presentations. For instance,
PL_std(4) = [[4] , [3,1] , [2,2] , [2,1,1] , [1,1,1,1]].
More about these PL-functions in the next section.

The number of the integer partitions will be denoted by P. So we have, for instance, P(4) = 5.
The function P will be treated in section 2.2.

2.1 Integer partitions as lists

We want to write a program giving a list of the partitions of an integer n. This means:
- Formulating an algorithm for obtaining the partitions.
- Transferring the algorithm into a computer program; in this case into a Maxima program.

2.1.1 Some heuristics by example

Almost every algorithm is based on a simple idea.

If we have a seemingly intractable problem, it is a good strategy, to reduce the given case to "smaller" cases, if possible. For example, instead of finding the partitions of 8, first find those of 7, then those of 6, and so on.
The small cases (partitions of 1, 2 and 3) can be seen right away. They are:
[[3], [2, 1] [1, 1, 1]] for n=3
[[2], [1, 1]] for n=2
[[1]] for n=1

Often this strategy means detecting a "recursive" structure in the problem. In the case of integer partitions the recursive idea is as follows:
In order to get the integer partitions of n, we take all the integer partitions of (n-1) and append the integer 1 to each of these partition-lists. The sum of each of these new lists is, of course, n. So, by this process, we get some, but not all of the partitions of n.

A good problem-solving strategy in almost all cases is to take a good, typical example and have a close look at it.

In this case, we let n = 8. The partitions of 8 are
[[8], [7,1], [6,2], [5,3], [4,4], [6,1,1], [5,2,1], [4,3,1], [4,2,2], [3,3,2], [5,1,1,1],
[4,2,1,1], [3,3,1,1], [3,2,2,1], [2,2,2,2],
[4,1,1,1,1], [3,2,1,1,1], [2,2,2,1,1], [3,1,1,1,1,1], [2,2,1,1,1,1], [2,1,1,1,1,1,1],
[1,1,1,1,1,1,1,1]]

In a more structured display (by sorting the partitions by the number of parts) we have for the list of partitions of 8:
[
[8],
[7,1], [6,2], [5,3], [4,4],
[6,1,1], [5,2,1], [4,3,1], [4,2,2], [3,3,2],
[5,1,1,1], [4,2,1,1], [3,3,1,1], [3,2,2,1], [2,2,2,2],
[4,1,1,1,1], [3,2,1,1,1], [2,2,2,1,1],
[3,1,1,1,1,1], [2,2,1,1,1,1],
[2,1,1,1,1,1,1],
[1,1,1,1,1,1,1,1]
]

The example shows that there are 22 integer partitions of 8.

According to the recursive strategy, we take a look at the partitions of 7. These are in the structured display:
[
[7],
[6,1], [5,2], [4,3],
[5,1,1], [4,2,1], [3,3,1], [3,2,2],
[4,1,1,1], [3,2,1,1], [2,2,2,1],
[3,1,1,1,1], [2,2,1,1,1],
[2,1,1,1,1,1],
[1,1,1,1,1,1,1]
]

By inserting (appending) 1 at the end of the partition-lists of 7 we obviously get a sublist of the partitions of 8:
[
[7, 1],
[6,1,1], [5,2,1], [4,3,1],
[5,1,1,1], [4,2,1,1], [3,3,1,1], [3,2,2,1],
[4,1,1,1,1], [3,2,1,1,1], [2,2,2,1,1],
[3,1,1,1,1,1], [2,2,1,1,1,1],
[2,1,1,1,1,1,1],
[1,1,1,1,1,1,1,1]
]

But we haven't got all of the partitions of 8. The following are missing:
Diff_0 := [[8], [6,2], [5,3], [4,4], [4,2,2], [3,3,2], [2,2,2,2]]

How do we get the missing partitions, preferably, in order to apply recursion, by somehow reducing the cases?
One way of reducing the cases in Diff_0 is to subtract 1 "everywhere". We thus arrive at:
Diff_1 := [[7], [5,1], [4,2], [3,3], [3,1,1], [2,2,1], [1,1,1,1]]

Obviously, if we add 1 to each of the intgers in Diff_1 (on the right-hand-side), we get exactly Diff_0 .
In a more structured display, Diff_1 looks like
[
[7], partitions of 7 split into 1 part
[5,1], [4,2], [3,3], partitions of 6 split into 2 parts
[3,1,1], [2,2,1], partitions of 5 split into 3 parts
[1,1,1,1] partitions of 4 split into 4 parts
]

Looking at the partitioning in this structured way, i.e. sorting the partitions according to number of parts, turns out to be a fruitful idea. In the following, we will denote by PL(n,k) the number of the integer partitions of n consisting of exactly k parts.

With this notation we can formulate the following strategy.

Strategy:
In order to construct the list PL(n, k) of the partitions of the integer n:
- construct the list PL(n-1, k-1) of the partitions of the integer n-1, and insert a 1 at the end of each sublist.
- for each k (1 < k < n) construct the list PL(n-k, k) of partitions of n-k with exactly k parts, and add 1 to each of the thus generated numbers.

Thus, the full list of partitions PL_full(n) of n decomposes into the lists PL(n-1, k-1) and PL(n-k, k).

```
In the above example with n=8 we get the lists
(1.)   Sublist_1  =  partitions of  7  (=8-1), augmented by 1:
[
[7, 1],                                          (= PL(7,1) augmented by 1 )
[6,1,1], [5,2,1], [4,3,1],                       (= PL(7,2) augmented by 1 )
[5,1,1,1], [4,2,1,1], [3,3,1,1], [3,2,2,1],      (= PL(7,3) augmented by 1 )
[4,1,1,1,1], [3,2,1,1,1], [2,2,2,1,1],           (= PL(7,4) augmented by 1 )
[3,1,1,1,1,1], [2,2,1,1,1,1],                    (= PL(7,5) augmented by 1 )
[2,1,1,1,1,1,1],                                 (= PL(7,6) augmented by 1 )
[1,1,1,1,1,1,1,1]                                (= PL(7,7) augmented by 1 )
]

and

(2.) Sublist_2  =  partitions of  8-k  into k parts, "list-added" by 1 (for 1 < k < 8):
[
[6,2], [5,3], [4,4], (= PL(6,2) with list-added 1; PL(6,2) = [ [5,1], [4,2], [3,3] ] )
[4,2,2], [3,3,2],    (= PL(5,3) with list-added 1; PL(5,3) = [ [3,1,1] , [2,2,1]] )
[2,2,2,2]            (= PL(4,4) with list-added 1; PL(4,4) = [ [1,1,1,1] ] )
]

and finally

(3.)  Sublist_3 =  partitions of  8  into 1 part:
[ [8] ]
```

Sublist_1 and Sublist_2 have no common elements, since each list in Sublist_1 has 1 as its last element, while the last elements in the lists of Sublist_2 are greater than 1.

Sublist_1 comprises 15 cases, Sublist_2 comprises 6 and Sublist_3 comprises 1 case.
Altogether, we have 22 integer partitions of 8.

When implementing the algorithm in a general form we will need some very simple auxiliary "helper"-functions discribed below. What they do should be clear from the descriptions above - and from their names and the respective comments.

```
insert1atend(L) := append(L, [1])  $
    /*  inserts a 1  at the end of the list L */
insert1atendall(LL) := map(insert1atend, LL)  $
     /*  inserts a  1  at the end of each list the list LL */
plus1(x)  := x+1  $
    /* raises the number x by 1  */
plus1list(L)  := map(plus1, L)  $
    /* raises each number in the list L by one  */
```

In the following algorithm we pull everything together.

```
PL(n, k)  :=
  if n<1 then [ ]  else
    if k<1 then [ ] else
      if k>n then [ ] else
        if n=1  then [ [1] ]   else
          if k=1 then   [ [n] ]  else
             append(insert1atendall(PL(n-1, k-1)), plus1list(PL(n-k, k)) ) ;
```

PL(n,k):=if n<1 then [] else if k<1 then [] else if k>n then [] else if n=1 then [[1]] else if k=1 then [[n]] else append(insert1atendall(PL(n-1,k-1)),plus1list(PL(n-k,k)))

```
for k:1 thru 8 do print("k =", k, ": ", PL(8,k));
```

k = 1 : [[8]]
k = 2 : [[7,1],[6,2],[5,3],[4,4]]
k = 3 : [[6,1,1],[5,2,1],[4,3,1],[4,2,2],[3,3,2]]
k = 4 : [[5,1,1,1],[4,2,1,1],[3,3,1,1],[3,2,2,1],[2,2,2,2]]
k = 5 : [[4,1,1,1,1],[3,2,1,1,1],[2,2,2,1,1]]
k = 6 : [[3,1,1,1,1,1],[2,2,1,1,1,1]]
k = 7 : [[2,1,1,1,1,1,1]]
k = 8 : [[1,1,1,1,1,1,1,1]]   done

The next function P_full just puts together all the PL(n,k)'s in a big list.

```
PL_full(n) := makelist(PL(n, k), k, 1, n);
```

PL_full(n):=makelist(PL(n,k),k,1,n)

```
PL_full(6);
```

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

Although this result contains all the necessary information, the standard form of presenting the list of partitions in this case is:

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

In order to obtain this, we have to remove the outer brackets of the list elements of PL_full. This process is called "flattening" in most Computeralgebra Systems. There is also a built-in function in Maxima called "flatten".

However: The Maxima function "flatten" removes brackets, but it removes all of them and this is not what we want in this case.

So we write our own function for flattening lists stepwise.
We call this function flattn(L,s). L is supposed to be a list and s (for steps) is an integer for denoting the number of flattening steps.
(It is recommended to try out this function on your own.)

```
flattn(L, s) :=           /* removes the outer brackets successively by s steps if possible */
  if s=0 then L else
    if L=[] then L else
      if not(listp(first(L))) then  append([first(L)], flattn(rest(L),s) )   else
       append(flattn(first(L), s-1), flattn(rest(L), s) )   $

/*  alternative:   xreduce('append, [ [[4]] , [[3,1],[2,2]] , [[2,1,1]] , [[1,1,1,1]] ] );    */
```

```
flattn(PL_full(6), 1);
```

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

In order to get the partitions in the standard form with one simple function call we define:

```
PL_std(n) := flattn(PL_full(n), 1);
```

PL_std(n):=flattn(PL_full(n),1)

```
PL_std(6);
```

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

This is what we wanted.

- -

Some tests

```
Test(n_test) :=
block(
print(length(PL_std(n_test) ) ),
print(sum(length(PL(n_test, k)), k, 1, n_test) ),
print(sum(length(PL(n_test-1, k-1)), k, 1, n_test) ),
print(sum(length(PL(n_test - k, k)), k, 1, n_test) ) )  ;
```

$$\mathrm{Test}(n_{test}):=\mathrm{block}\left(\mathrm{print}(\mathrm{length}(\mathrm{PL\_std}(n_{test}))),\mathrm{print}\left(\sum_{k=1}^{n_{test}}(\mathrm{length}(\mathrm{PL}(n_{test},k)))\right),\mathrm{print}\left(\sum_{k=1}^{n_{test}}(\mathrm{length}(\mathrm{PL}(n_{test}-1,k-1)))\right),\mathrm{print}\left(\sum_{k=1}^{n_{test}}(\mathrm{length}(\mathrm{PL}(n_{test}-k,k)))\right)\right)$$

```
Test(45) $;
```

89134
89134
75175
13959

- -

Finally, in this section, the following two "helper" functions used in the above text.

```
SetDiff : setdifference(setify(PL_std(8)), setify(insert1atendall(PL_std(7)) ) ) ;
```

$\langle$[2,2,2,2],[3,3,2],[4,2,2],[4,4],[5,3],[6,2],[8]$\rangle$

```
tex(%);                    /*   conversion to TeX   */
```

```
$$\left \{\left[ 2 , 2 , 2 , 2 \right]  , \left[ 3 , 3 , 2 \right]
 , \left[ 4 , 2 , 2 \right]  , \left[ 4 , 4 \right]  , \left[ 5 , 3
 \right]  , \left[ 6 , 2 \right]  , \left[ 8 \right]  \right \}$$       false
```

```
sort(listify(SetDiff), ordergreatp) ;
```

[[8],[6,2],[5,3],[4,4],[4,2,2],[3,3,2],[2,2,2,2]]

2.1.2 Maxima's built-in partition function

In Maxima, there is a built-in function integer_partitions giving essentially the same result as in section 1.3.1. But the result is given as a set sorted in a different order. In the following few commands these results are compared.

```
integer_partitions(8);        /* built-in function:  the result is a set */
```

{[1,1,1,1,1,1,1,1],[2,1,1,1,1,1,1],[2,2,1,1,1,1],[2,2,2,1,1],[2,2,2,2],[3,1,1,1,1,1],[3,2,1,1,1],[3,2,2,1],[3,3,1,1],[3,3,2],[4,1,1,1,1],[4,2,1,1],[4,2,2],[4,3,1],[4,4],[5,1,1,1],[5,2,1],[5,3],[6,1,1],[6,2],[7,1],[8]}

```
length(%);
```

22

```
listify(integer_partitions(8));    /* now the result is a list */
```

```
[[1,1,1,1,1,1,1,1],[2,1,1,1,1,1,1],[2,2,1,1,1,1],[2,2,2,1,1],[2,2,2,2],
[3,1,1,1,1,1],[3,2,1,1,1],[3,2,2,1],[3,3,1,1],[3,3,2],[4,1,1,1,1],[4,2,1,1],[4,2,2],
[4,3,1],[4,4],[5,1,1,1],[5,2,1],[5,3],[6,1,1],[6,2],[7,1],[8]]
```

```
length(%);
```

```
22
```

```
sort(listify(integer_partitions(8)), ordergreatp);    /* result sorted in descending order */
```

```
[[8],[7,1],[6,2],[6,1,1],[5,3],[5,2,1],[5,1,1,1],[4,4],[4,3,1],[4,2,2],
[4,2,1,1],[4,1,1,1,1],[3,3,2],[3,3,1,1],[3,2,2,1],[3,2,1,1,1],[3,1,1,1,1,1],[2,2,2,2]
,[2,2,2,1,1],[2,2,1,1,1,1],[2,1,1,1,1,1,1],[1,1,1,1,1,1,1,1]]
```

```
PL_std(8);               /* for comparison  */
```

```
[[8],[7,1],[6,2],[5,3],[4,4],[6,1,1],[5,2,1],[4,3,1],[4,2,2],[3,3,2],[5,1,1,1],
[4,2,1,1],[3,3,1,1],[3,2,2,1],[2,2,2,2],[4,1,1,1,1],[3,2,1,1,1],[2,2,2,1,1],
[3,1,1,1,1,1],[2,2,1,1,1,1],[2,1,1,1,1,1,1],[1,1,1,1,1,1,1,1]]
```

```
sort(PL_std(8), ordergreatp);
```

```
[[8],[7,1],[6,2],[6,1,1],[5,3],[5,2,1],[5,1,1,1],[4,4],[4,3,1],[4,2,2],
[4,2,1,1],[4,1,1,1,1],[3,3,2],[3,3,1,1],[3,2,2,1],[3,2,1,1,1],[3,1,1,1,1,1],[2,2,2,2]
,[2,2,2,1,1],[2,2,1,1,1,1],[2,1,1,1,1,1,1],[1,1,1,1,1,1,1,1]]
```

```
 /* comparing the results */

is(sort(PL_std(8), ordergreatp) = sort(listify(integer_partitions(8)), ordergreatp) );
```

```
true
```

Question: What is the reason for writing one's own partition function?

It is possible to use built-in functions without knowing how they work. In many cases this is OK.

But in order to understand a particular problem and its solution, it is indispensible to know in detail how this solution works. In algorithmic mathematics this means: You have to fully understand the workings of the pertaining algorithm and be able to write a program solving the problem on the basis of this algorithm.

2.2 The number of partitions of an integer

As seen above, the number of integer partitons of 8 is 22.

```
length(PL_std(8));                 /*  confirmation */
```

```
22
```

If we are only interested in the _number_ of partitions, there is no need to generate a list of all the partitions and then count them by applying the length function.
Instead we can directly compute the number of partitions recursively by using more or less the same recursion structure as in the Maxima function PL above.

At the core of the recursion is the equation   P(n, k) = P(n-1, k-1) + P(n-k, k)

In Maxima, this recursion, in analogy to PL, is implemented by the following program based on the number k of the parts (Ps stands for partitions with fixed number s of parts)

```
Ps(n, k)  :=                   /*  Ps(n,k):  the number of all partitions */
    if n<1 then 0  else          /*  of n consisting of exactly k parts  */
       if k<1 then 0  else           /* explanation:  Ps for P stepwise */
         if k>n then 0  else
              if k=1 then 1 else
               Ps(n-1, k-1) + Ps(n-k, k)     ;
```

```
Ps(n,k):=if n<1 then 0 else if k<1 then 0 else if k>n then 0 else if k=1 then 1 else
Ps(n-1,k-1)+Ps(n-k,k)
```

```
P(n)  := sum(Ps(n,k), k, 1, n);
```

$$P(n) := \sum_{k=1}^{n} (Ps(n,k))$$

```
P(45);
```

```
89134
```

```
/* a test */
for i:0 thru 9 do print(i, Ps(8,i), length(PL(8,i)) ) ;
```

```
0  0  0
1  1  1
2  4  4
3  5  5
4  5  5
5  3  3
6  2  2
7  1  1
8  1  1
9  0  0         done
```

```
/* another test */
for i:1 thru 20 do print(i, P(i), length(PL_std(i)) ) ;
```

```
1  1  1
2  2  2
3  3  3
4  5  5
5  7  7
6  11  11
7  15  15
8  22  22
```

```
9  30  30
10  42  42
11  56  56
12  77  77
13  101  101
14  135  135
15  176  176
16  231  231
17  297  297
18  385  385
19  490  490
20  627  627        done
```

The values of the function P are rising extremely fast. So, for instance, with the "extended" stamping problem (an unlimited supply of stamps exists for any number between 1 and 45) we have:

```
P(45);
```

89134

The functions PL and P, being fully recursive, will become extremely slow in evaluating their arguments. Im Maxima, like in other Computer Algebra Systems (CAS) there are methods of speeding up the evaluation of fully recursive functions. One of them, which is applicable in special cases, is "tail recursion". It is not applicable, here, and we will not treat it, here.
A very general and easy to apply method of dealing with this problem is known as "memoizing functions" or "array functions".

This method is in a nutshell: full recursion is extremely slow, because in the recursion process there are many calls of the function with smaller arguments - and these intermediate values are "forgotten" after the call is finished.   When a function f is defined as a memoizing function, every value of f that is ever computed, is stored in an array table, and when this value is needed again, it is not computed any more but looked up in the array. With fully recursive functions, this is much faster.

Of course there is a price to this. Instead of time, now space (for the array) is needed. So, there is a tradeoff problem, depending on the specific problem. And each user has to decide about how to handle the tradeoff.

The syntax for defining array-functions in Maxima is (observe the different brackets / parentheses):

f[x] := ... (followed by a Maxima expression)
instead of
f(x) := ... (followed by a Maxima expression)

The following function PAs (A for array) is a much faster version of the recursive Ps function.
When PAs[n,k]   is called with a high value for n (like n=1000) for the first time, it needs some time to build up the array. With further calls with similarly high values of n the result is delivered more or less instantaneously.
(More on time measurement is to be found in the worksheet "Fib-timing.wxmx".)

```
PAs[n, k] :=
    if n<1 then 0  else
       if k<1 then 0  else
         if k>n then 0  else
              if k=1 then 1 else
               PAs[n-1, k-1] + PAs[n-k, k]  ;
```

$PAs_{n,k} := \text{if } n<1 \text{ then } 0 \text{ else if } k<1 \text{ then } 0 \text{ else if } k>n \text{ then } 0 \text{ else if } k=1 \text{ then } 1 \text{ else } PAs_{n-1,k-1} + PAs_{n-k,k}$

```
PAs[45,3];
```

169

```
PAs[85,3] ;
```

602

```
for k:1 thru 8 do print("k =", k, ": ", PAs[8,k] ) ;
```

```
k = 1 :   1
k = 2 :   4
k = 3 :   5
k = 4 :   5
k = 5 :   3
k = 6 :   2
k = 7 :   1
k = 8 :   1        done
```

```
 /*  PA[n]:  the number of all partitions as an array function  */

PA[n]   := sum(PAs[n,k], k, 1, n)

/* alternative
PA[n] :=
  block( [P1],
   P1 : 0,
   for k:1 thru n do  P1 : P1 + PAs[n, k],
   return(P1) );
*/;
```

$$PA_n := \sum_{k=1}^{n} (PAs_{n,k})$$

```
 PA[45];
```

89134

```
PA[85];
```

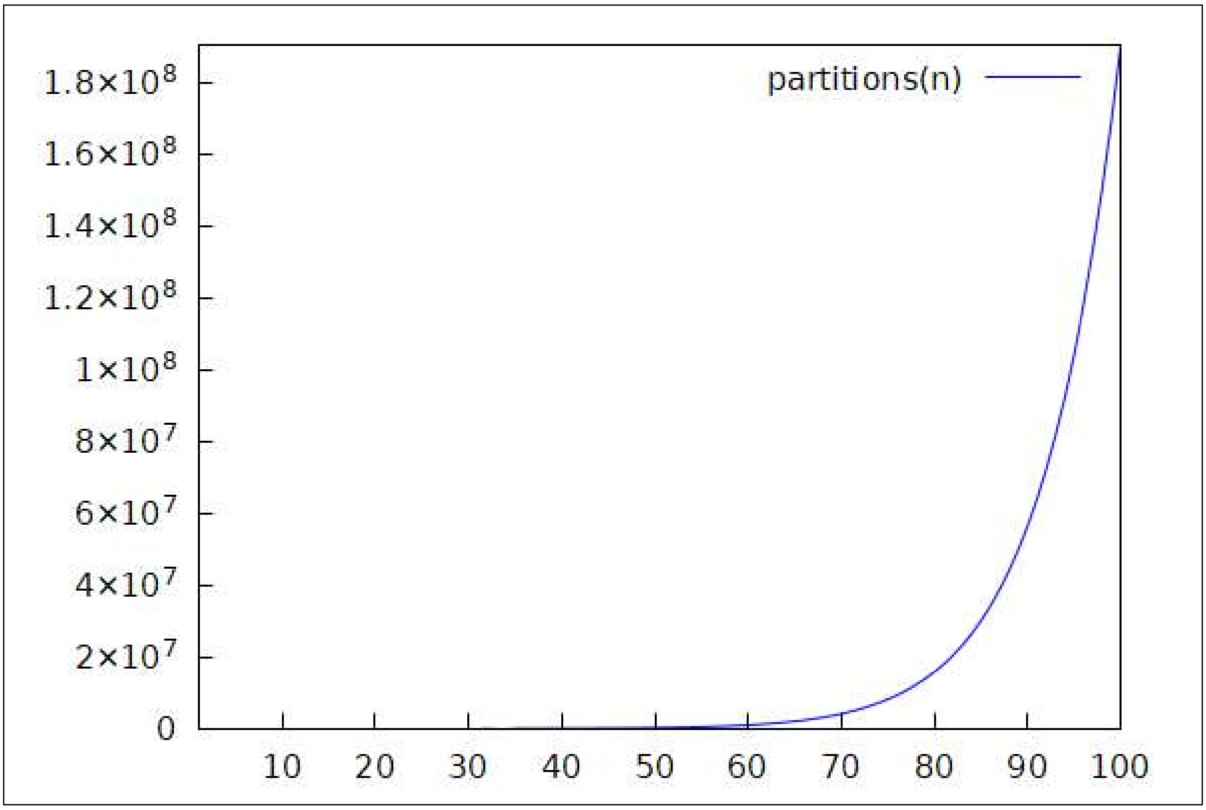
30167357

```
PA[1000];           /* P(1000) would take "forever" */  ;
```

```
        24061467864032622473692149727991
max : 100 $
L1 :  makelist(PA[n], n, 1, max) $
G1 : [ key="partitions(n)",   color=blue, point_size=0, points_joined=true, points(L1) ]  $
makelist(PA[n], n, 1, max);
```

```
        [1,2,3,5,7,11,15,22,30,42,56,77,101,135,176,231,297,385,490,627,792,1002,1255,1575,
1958,2436,3010,3718,4565,5604,6842,8349,10143,12310,14883,17977,21637,26015,31185,37338,44583,
53174,63261,75175,89134,105558,124754,147273,173525,204226,239943,281589,329931,386155,451276,
526823,614154,715220,831820,966467,1121505,1300156,1505499,1741630,2012558,2323520,2679689,3087735,
3554345,4087968,4697205,5392783,6185689,7089500,8118264,9289091,10619863,12132164,13848650,15796476
,18004327,20506255,23338469,26543660,30167357,34262962,38887673,44108109,49995925,56634173,64112359
,72533807,82010177,92669720,104651419,118114304,133230930,150198136,169229875,190569292]
```

```
wxdraw2d(G1) ;
```



As we have seen, the stamping problem can be thought of as being part of the partitioning problem in the following sense. The results of stamping and partitioning are the same, if there is an unlimited supply of stamps of every value, i.e. an unlimited supply of 1-Ct, 2-Ct, 3-Ct, ... , n-Ct stamps.

3 The triangle of partition numbers

In this section we refer to https://en.wikipedia.org/wiki/Triangle_of_partition_numbers

The following function gives the numbers of the integer partitions of n as a list of these numbers consisting of 1, 2, 3, ..., n parts, their sum, of course, being P(n).

```
Ps_Tri(n) := makelist(Ps(n,i), i, 1, n);
```

```
        Ps_Tri(n):=makelist(Ps(n,i),i,1,n)
```

```
Ps_Tri(9);
```

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

```
for n:1 thru 9 do print("n =", n, ": ", Ps_Tri(n)) ;
```

```
n = 1 :  [1]
n = 2 :  [1,1]
n = 3 :  [1,1,1]
n = 4 :  [1,2,1,1]
n = 5 :  [1,2,2,1,1]
n = 6 :  [1,3,3,2,1,1]
n = 7 :  [1,3,4,3,2,1,1]
n = 8 :  [1,4,5,5,3,2,1,1]
n = 9 :  [1,4,7,6,5,3,2,1,1]        done
```

The purpose of the next commands is better formatting

```
PT9 : makelist(append(["n  =", n, ":"], Ps_Tri(n)), n, 1, 9);
```

```
        [[n  =,1,:,1],[n  =,2,:,1,1],[n  =,3,:,1,1,1],[n  =,4,:,1,2,1,1],
[n  =,5,:,1,2,2,1,1],[n  =,6,:,1,3,3,2,1,1],[n  =,7,:,1,3,4,3,2,1,1],
[n  =,8,:,1,4,5,5,3,2,1,1],[n  =,9,:,1,4,7,6,5,3,2,1,1]]
```

```
PT20 : makelist(append(["n  = ", n, ":"], Ps_Tri(n)), n, 1, 20);
```

```
        [[n  = ,1,:,1],[n  = ,2,:,1,1],[n  = ,3,:,1,1,1],[n  = ,4,:,1,2,1,1],
[n  = ,5,:,1,2,2,1,1],[n  = ,6,:,1,3,3,2,1,1],[n  = ,7,:,1,3,4,3,2,1,1],
```

```
[n = ,8,:,1,4,5,5,3,2,1,1],[n = ,9,:,1,4,7,6,5,3,2,1,1],
[n = ,10,:,1,5,8,9,7,5,3,2,1,1],[n = ,11,:,1,5,10,11,10,7,5,3,2,1,1],
[n = ,12,:,1,6,12,15,13,11,7,5,3,2,1,1],[n = ,13,:,1,6,14,18,18,14,11,7,5,3,2,1,1],
[n = ,14,:,1,7,16,23,23,20,15,11,7,5,3,2,1,1],
[n = ,15,:,1,7,19,27,30,26,21,15,11,7,5,3,2,1,1],
[n = ,16,:,1,8,21,34,37,35,28,22,15,11,7,5,3,2,1,1],
[n = ,17,:,1,8,24,39,47,44,38,29,22,15,11,7,5,3,2,1,1],
[n = ,18,:,1,9,27,47,57,58,49,40,30,22,15,11,7,5,3,2,1,1],
[n = ,19,:,1,9,30,54,70,71,65,52,41,30,22,15,11,7,5,3,2,1,1],
[n = ,20,:,1,10,33,64,84,90,82,70,54,42,30,22,15,11,7,5,3,2,1,1]]
```

```
table_form(PT20)$          /* table_form:  non-standard command of wxMaxima */
```

```
n =  1 : 1
n =  2 : 1 1
n =  3 : 1 1 1
n =  4 : 1 2 1 1
n =  5 : 1 2 2 1 1
n =  6 : 1 3 3 2 1 1
n =  7 : 1 3 4 3 2 1 1
n =  8 : 1 4 5 5 3 2 1 1
n =  9 : 1 4 7 6 5 3 2 1 1
n = 10 : 1 5 8 9 7 5 3 2 1 1
n = 11 : 1 5 10 11 10 7 5 3 2 1 1
n = 12 : 1 6 12 15 13 11 7 5 3 2 1 1
n = 13 : 1 6 14 18 18 14 11 7 5 3 2 1 1
n = 14 : 1 7 16 23 23 20 15 11 7 5 3 2 1 1
n = 15 : 1 7 19 27 30 26 21 15 11 7 5 3 2 1 1
n = 16 : 1 8 21 34 37 35 28 22 15 11 7 5 3 2 1 1
n = 17 : 1 8 24 39 47 44 38 29 22 15 11 7 5 3 2 1 1
n = 18 : 1 9 27 47 57 58 49 40 30 22 15 11 7 5 3 2 1 1
n = 19 : 1 9 30 54 70 71 65 52 41 30 22 15 11 7 5 3 2 1 1
n = 20 : 1 10 33 64 84 90 82 70 54 42 30 22 15 11 7 5 3 2 1 1
```

It is illustrative, and highly recommended, to display the relation   P(n, k) = P(n-1, k-1) + P(n-k, k)   in this diagram.

4 Other methods

4.1 Memoizing functions / array functions

see functions PA[n] and PAs[n,k] above

4.2 Approximation

Ramanujan-Hardy:

```
P_approx(n) := 1/(4*n*sqrt(3)) * exp(%pi * sqrt(2*n/3) ) ;
```

$$\mathrm{P\_approx}(n) := \frac{1}{4*n*\sqrt{3}} * \exp\left(\pi * \sqrt{\frac{2*n}{3}}\right)$$

```
A1000 : floor(float(P_approx(1000) ) );
```

24401996316803111446873727041536

```
PA[1000]              ;/*  mind the square brackets !  */
```

24061467864032622473692149727991

```
A1000 - PA[1000] ;
```

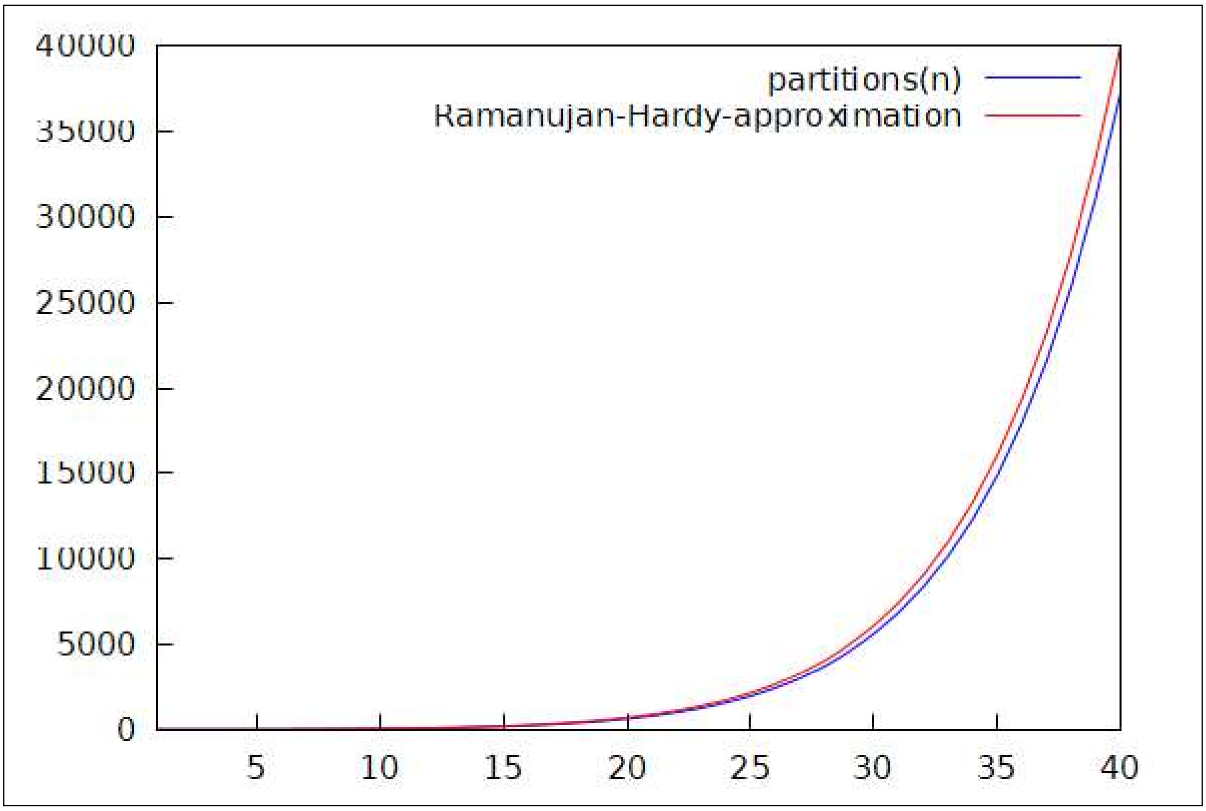
340528452770488973181577313545

```
float(A1000 / PA[1000]);      /*  1.415 percent too large */
```

1.0141524388576275

```
max : 40 $
L1 :  makelist(PA[n], n, 1, max) $
L2 :  makelist(float(P_approx(n)), n, 1, max) $
G1 : [ key="partitions(n)",   color=blue, point_size=0, points_joined=true, points(L1) ]  $
G2 : [ key="Ramanujan-Hardy-approximation", color=red,  point_size=0, points_joined=true, points(L2) ]  $
```

```
wxdraw2d( G1, G2 ) ;
```



5 Ramanujan

Srinivasa Ramanujan (1887 - 1920) was one of India's greatest mathematical geniuses. Though he had almost no formal training in pure mathematics, he made substantial contributions to mathematical analysis, number theory, infinite series, and continued fractions, including solutions to mathematical problems then considered unsolvable.

Ramanujan also worked on partitions intensely and in cooperation with the great British mathematician G.H. Hardy (1877-1947) found the approximating function above (P_approx(n)) in 1918, see: https://en.wikipedia.org/wiki/Partition_function_(number_theory).

Srinivasa Ramanujan discovered that the partition function has nontrivial patterns in modular arithmetic, now known as Ramanujan's congruences. For instance, whenever the decimal representation of n ends in the digit 4 or 9, the number of partitions of n will be divisible by 5.

Literature and media on Ramanujan:

Kanigel R.: The Man who knew Infinity. New York: Washington Square Press, 1991
    German translation: A. Beutelspacher: Der das Unendliche kannte, Braunschweig/Wiesbaden 1995
    Film ("trailer"): Die Poesie des Unendlichen: https://www.imdb.com/title/tt0787524/

Video with A. Beutelspacher:
https://www.youtube.com/watch?v=4x3qXRgnYgw&list=PLfdMKJMGPPtw7kmxHU1e4F-GRiqsJi3L3&index=1

https://en.wikipedia.org/wiki/Srinivasa_Ramanujan
  German: https://de.wikipedia.org/wiki/Srinivasa_Ramanujan

https://mathshistory.st-andrews.ac.uk/Biographies/Ramanujan/

https://www.mathematik.de/Vermischtes/112-wer-war-eigentlich-srinivasa-ramanujan