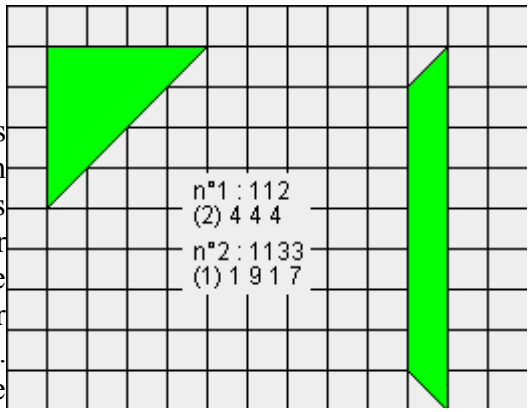
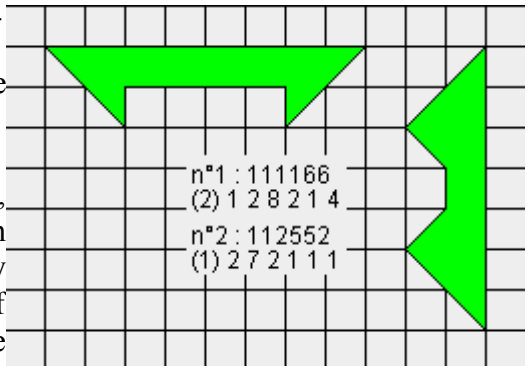
1) Different ways to digitalize a shape of the tan's family

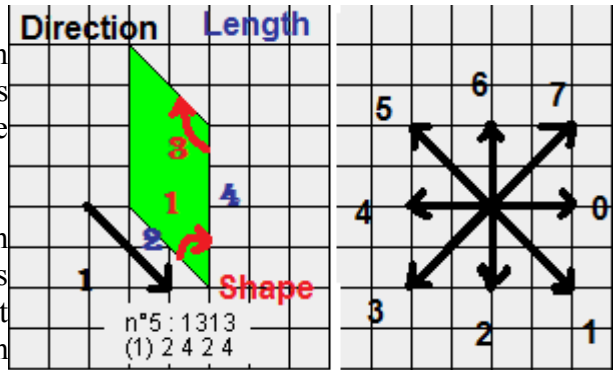
a) The first way: shape and length numbers



2 strings "shapeNumber" and "lengthNumber".
This codification come from Eric's program.
For the shape number, the codification of the inside angles are : 1 for 45°, 2 for 90°, 3 for 135°, 5, 6 or 7. The length number tell if there is 1, 2, 3, ... length units. Length units have not the same value if the direction is along the grid or in the diagonal direction. These strings are used in the array-form S[] and L[] (the S for Shape and the L for Length). Additionnaly the array D[] is used for Directions. The first direction digit D[0] is the most important one (other will be computed from it) and is equal to the L[0]. So L[1] is the first length digit, S[0]is the first shape digit.
If there is C sides, the last shape digit will be S[C-1], the last length digit L[C], and the last direction digit D[C-1].

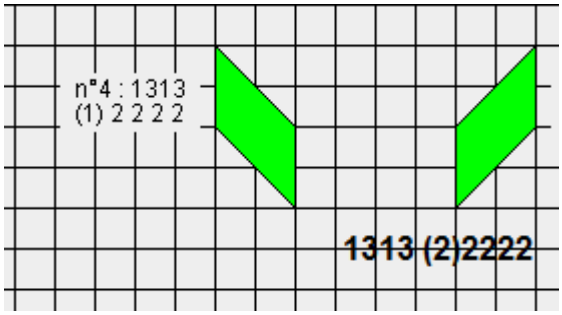


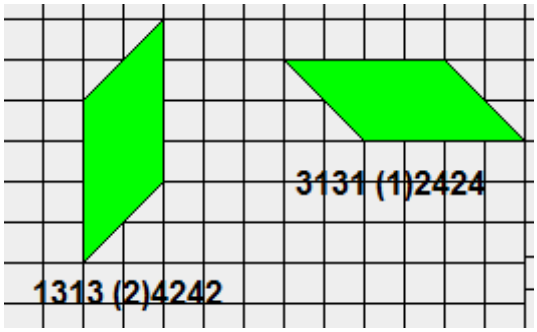
In the picture some exemples : the length number first, then the L[0]=D[0] between brackets, and then the length number. The first direction digit (L[0]=D[0]) could be only 1 or 2 : 1 if the first side is in the diagonal direction, 2 if the first side is in the vertical direction (see the 8 possible direction digits (0,1,2,3,4,5,6,7) in the picture).



Shape numbers and length numbers are computed in order to classify them as integer numbers (112 is before 1133 which is before 111166 which is before 112552, ...)

In order to avoid that the same tan appears with different codifications, the first one is kept, others are deleted or ignored. For instance, this tan is kept with this number 1313 12424. But it could appear in other digitizations : 3131 12424 or also 1313 24242.
Seel also the tan 1313 12222 which could appear with same length and shape numbers, the first direction digit only makes the difference.





b) The cartesian coordinates

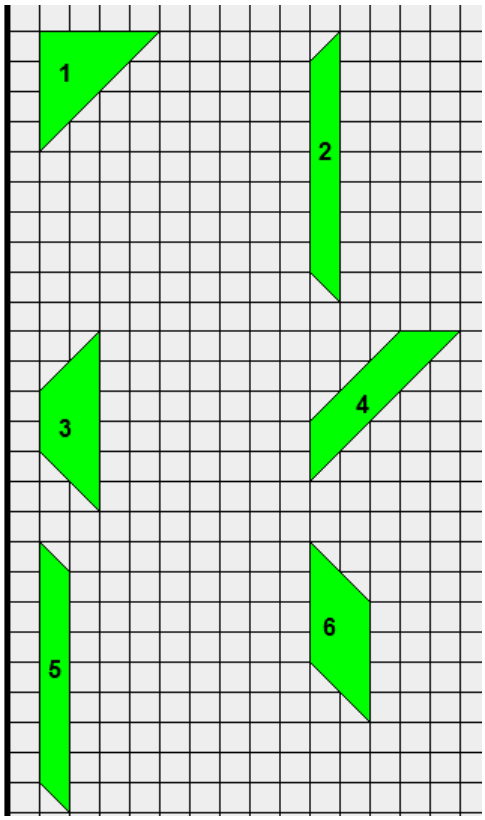
These are usefull for the drawing of the polygons (java command of drawing a polygon neads this arrays of coordinates). There are easy to comput from the 3 arrays of digits S, L and D. We start from x[0]=0 and y[0]=0 and comput first x[1] and y[1] using both D[0] and L[1], then we advance and comput x[2] and y[2] using both D[1] and L[2], ...

For instance, the big triangle of 16 triangles area is digitized by the following coordinates :

- x[0]=y[0]=0
- x[1]=0, y[1]=4
- x[2]=4, y[2]=0
- x[3]=y[3]=0

The positive value of y are in the down direction. See this printing of the program for other exemples.

```
n:1 shapeNumber=112
x[0]=0 y[0]=0
x[1]=0 y[1]=4
x[2]=4 y[2]=0
n:2 shapeNumber=1133
x[0]=0 y[0]=8
x[1]=1 y[1]=9
x[2]=1 y[2]=0
x[3]=0 y[3]=1
n:3 shapeNumber=1133
x[0]=0 y[0]=4
x[1]=2 y[1]=6
x[2]=2 y[2]=0
x[3]=0 y[3]=2
n:4 shapeNumber=1133
x[0]=0 y[0]=3
x[1]=0 y[1]=5
x[2]=5 y[2]=0
x[3]=3 y[3]=0
n:5 shapeNumber=1313
x[0]=0 y[0]=8
x[1]=1 y[1]=9
x[2]=1 y[2]=1
x[3]=0 y[3]=0
n:6 shapeNumber=1313
x[0]=0 y[0]=4
x[1]=2 y[1]=6
x[2]=2 y[2]=2
x[3]=0 y[3]=0
n:7 shapeNumber=1313
x[0]=0 y[0]=2
x[1]=4 y[1]=6
x[2]=4 y[2]=4
x[3]=0 y[3]=0
n:8 shapeNumber=1313
x[0]=0 y[0]=1
x[1]=8 y[1]=9
x[2]=8 y[2]=8
x[3]=0 y[3]=0
n:9 shapeNumber=1223
x[0]=0 y[0]=3
x[1]=2 y[1]=5
x[2]=2 y[2]=0
x[3]=0 y[3]=0
n:10 shapeNumber=1223
```



When the shapes are computed, it is a way to recognize polygons (closed lines) from open lines : if C is the number of sides x[C] and y[C] must be equal to x[0] and y[0], says to 0. These coordinates are also used to recognize polygons with cross sides, or polygons wich have a vertex on a side.

The coordinates are also multiplied in order to draw the polygons. We are using a square grid of 20 pixels for 1 unit. Then, when the tans are moved (in the 'play' mode) we translate the tans be changing the coordinates.

c) <u>The L1 number</u>

This number appear like a string of digits. Each digit has a meaning :

- 0 is for 1 length unit in the grid direction.
- 9 is for 1 length unit in the diagonal direction.
- 1,2,3,5,6,7 are the shape digits

For instance, the big triangle of 16 triangles area is digitized by a L1 number : 000019999100002
This means that you can draw this shape : 4 length units in the grid direction (either vertical or horizontal) then a 45° angle, then 4 length units in the diagonal direction and another 45° angle. The last angle is not necessary but still given. Here are the java console printing of what appears when you comput the different convex shapes with 16 triangles.

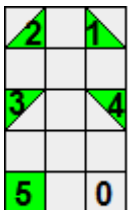
```
n:1 shapeNumber=112 lengthNumber=2 4 4 4  L1=000019999100002
n:2 shapeNumber=1133 lengthNumber=1 1 9 1 7  L1=00000000019300000000391
n:3 shapeNumber=1133 lengthNumber=1 2 6 2 2  L1=0000001993003991
n:4 shapeNumber=1133 lengthNumber=2 2 5 2 3  L1=0019999910039993
n:5 shapeNumber=1313 lengthNumber=1 1 8 1 8  L1=00000000193000000000193
n:6 shapeNumber=1313 lengthNumber=1 2 4 2 4  L1=0000199300001993
n:7 shapeNumber=1313 lengthNumber=1 4 2 4 2  L1=0019999300199993
n:8 shapeNumber=1313 lengthNumber=1 8 1 8 1  L1=01999999993019999999993
n:9 shapeNumber=1223 lengthNumber=1 2 5 2 3  L1=0000019930002002
n:10 shapeNumber=1223 lengthNumber=2 4 3 2 1  L1=00001999299293
n:11 shapeNumber=2222 lengthNumber=1 1 4 1 4  L1=29299992929999
n:12 shapeNumber=2222 lengthNumber=1 2 2 2 2  L1=299299299299
n:13 shapeNumber=2222 lengthNumber=2 1 8 1 8  L1=00000000202000000000202
n:14 shapeNumber=2222 lengthNumber=2 2 4 2 4  L1=0000200200002002
n:15 shapeNumber=13233 lengthNumber=1 3 3 2 1 1  L1=000199930392993
n:16 shapeNumber=22323 lengthNumber=2 1 4 1 2 2  L1=000020399299302
n:17 shapeNumber=223333 lengthNumber=2 2 3 2 1 1 1  L1=00020039303930002
n:18 shapeNumber=233233 lengthNumber=1 1 1 3 1 1 3  L1=00039293000392993
n:19 shapeNumber=233233 lengthNumber=2 1 2 2 1 2 2  L1=00203993002039993
n:20 shapeNumber=233233 lengthNumber=2 2 2 1 2 2 1  L1=00200393002003993
```

These number is not used in the Eric program. We have needed it in the process of deleting double shapes. At the begining, our rules where not strong enough, or too strong, so we delete not double shapes or keep double ones. With this number, ordered in the integer order, we have a unique signature

for each tan, easy to compare with others to kkep or delete new shapes.

d) <u>The last way to digitalize a tan : a 6 values array of arrays</u>

This is an array of arrays, let us say DT[j][i] where i is a column index and j is a row index. It gives, with integers from 0 to 5 (6 values), the place occupied by the inside of the tan, in a decomposition based on the grid. The codification used is the following :



- 0 is used when the tan is not located on this square of the grid
- 5 is used when the tan is fully located on this square of the grid
- 1, 2, 3 or 4 are used when only a half part of the square is occupied by the tan.

The picture shows this codification.

With this codification we get for the first convex shapes of 16 triangles the following :

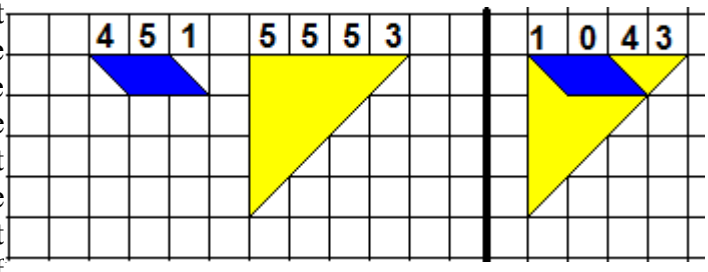
```
iP=15 shape=
 5 5 5 3
 5 5 3 0
 5 3 0 0
 3 0 0 0
iP=18 shape=
 2
 5
 5
 5
 5
 5
 5
 4
iP=15 shape=
 0 2
 2 5
 5 5
 5 5
 4 5
 0 4
iP=15 shape=
 0 0 2 5 3
 0 2 5 3 0
 2 5 3 0 0
 5 3 0 0 0
 3 0 0 0 0
iP=18 shape=
 1
 5
 5
 5
 5
 5
 5
 4
```

```
iP=15 shape=
 1 0
 5 1
 5 5
 5 5
 4 5
 0 4
iP=15 shape=
 1 0 0 0
 5 1 0 0
 4 5 1 0
 0 4 5 1
 0 0 4 5
 0 0 0 4
iP=18 shape=
 1 0 0 0 0 0 0 0
 4 1 0 0 0 0 0 0
 0 4 1 0 0 0 0 0
 0 0 4 1 0 0 0 0
 0 0 0 4 1 0 0 0
 0 0 0 0 4 1 0 0
 0 0 0 0 0 4 1 0
 0 0 0 0 0 0 4 1
 0 0 0 0 0 0 0 4
iP=15 shape=
 5 5
 5 5
 5 5
 4 5
 0 4
iP=14 shape=
 2 1 0
 5 5 1
 5 5 3
 5 3 0
 3 0 0
```

This codification has been used for the recorded shapes and the tans of a tanset in the research of solutions. To know if a tan fit a place, we just have to try to substract the digits of the tans from those of the shape with this special rule :

- It is possible to substract 0, 1, 2, 3 or 4 from 5 (5-4=1 ; 5-3=2 ; ..)
- It is possible to substract 0 from each digit (1-0=0 ; 2-0=0 ; ..)
- It is impossible to make other substraction : 4-1 has no solution (it is 4 which remains), the same for 3-2 or 3-1, ..

With these rules we just have to try to substract each digits, if a soubstraction is impossible we know that the tan doesn't fit, if it is possible we keep the digitized shape with the results of the substractions. For instance, if I need to try to fit the tan digitized 451 from the big triangle digitized 5553 on the first row. From the first position of the tan (at the upper left corner of



the shape) it is possible, and I get 1043 still remaining on the first raw to be fitted by the other tans. The shape is matched by all the tans if, at the end of the process (when all the tans of the tanset has been tried) we get only 0 for each digit of this DT[j][i] array.

Of course the digitization 451 is not the only one possible for the mentioned tan. It could be turn and flipped. For an ordinary tan (without special symmetry features) it could be digitized with 8 different arrays. This number is less for symmetric tans, for instance the small square of 2 triangles has only one digitization : 5. So, we are examining each tan to recognize its family of symmetry and then comput all the alias of the tans that would be needed for the process. Here is a printing that show the result for some tans from the Tangram set.

```
iP=4 tan no1=
 4
type de symetrie de la piece : 4
apres une symetrie horizontale, alias1=
 2
apres une symetrie verticale, alias2=
 3
apres un demi-tour, alias4=
 1
apres une rotation, alias3=
 2
apres une symetrie d'axe d2, alias5=
 1
apres une symetrie d'axe d1, alias6=
 4
apres une rotation inverse, alias7=
 3
```

```
iP=4 tan no3=
 1
 3
type de symetrie de la piece : 2
apres une symetrie horizontale, alias1=
 1
 3
apres une symetrie verticale, alias2=
 2
 4
apres un demi-tour, alias4=
 2
 4
apres une rotation, alias3=
 4 3
apres une symetrie d'axe d2, alias5=
 4 3
apres une symetrie d'axe d1, alias6=
 2 1
apres une rotation inverse, alias7=
 2 1
```

```
iP=4 tan no4=
 1
 4
type de symetrie de la piece : 3
apres une symetrie horizontale, alias1=
 2
 3
apres une symetrie verticale, alias2=
 2
 3
apres un demi-tour, alias4=
 1
 4
apres une rotation, alias3=
 2 3
apres une symetrie d'axe d2, alias5=
 4 1
apres une symetrie d'axe d1, alias6=
 4 1
apres une rotation inverse, alias7=
 2 3
```

```
iP=4 tan no5=
 5
type de symetrie de la piece : 9
apres une symetrie horizontale, alias1=
 5
apres une symetrie verticale, alias2=
 5
apres un demi-tour, alias4=
 5
apres une rotation, alias3=
 5
apres une symetrie d'axe d2, alias5=
 5
apres une symetrie d'axe d1, alias6=
 5
apres une rotation inverse, alias7=
 5
```

2) How does it works

a) To find all the shapes

The shape numbers are taken from 112 (the lesser shape number) to 33333333 if it is a convex shape. For each shape number we try each possible length numbers (starting at 111..), in the two possible ways of the first digit number (0 or 1).

In order to fasten the program we have put some rules that must be obeied by the digits. Some are very general, commun for all the shapes, and others are special but still very usefull. For instance lengthMax is the maximum possible value of the length digit for a shape. There are two rules used, one being predominant quickly :
lengthMax=triangles+3-cotes; if(triangles/2+1<lengthMax)lengthMax=triangles/2+1;
An exemple of very peticuliar rule : for the convex shapes of 7 sides, the length digits should obey to this rule : *L[1]=(2*L[4]+L[5]+L[3]-L[7])/2* and *L[2]=(2*L[6]+L[7]+L[5]-L[3])/2* are the 2 digit that could computed without trying other values. Other values are not possible. See the extract of the code that shows the rule for the 2 different possible first direction digit.

```
if(cotes==7){if(L[0]==1){L[1]=(2*L[4]+L[5]+L[3]-L[7])/2;L[2]=(2*L[6]+L[7]+L[5]-L[3])/2;}
        else    {L[1]=L[3]+L[4]+L[5]-L[7];    L[2]=L[5]+L[6]+L[7]-L[3];}
                if(L[1]<1||L[2]<1)impossible=true;}
```

Many of the rules used here are taken from the Eric's program. My preferred one is the rule that links the area, the vertice in the perimeter and the vertice inside the shape known as the Pick formula. This rule is not used like that but just to compare the perimeter and the number of trianles of the tan : there must have the same parity (both odd or both even), so we just ask if *perimeter+triangles* is an even number. For the different possible values of perimeter (sum of the length digits) we had these rules :
PerimeterMax=triangles+2 *PerimeterMin=(int)(Math.sqrt(4.0*triangles))*
The first one come from the Pick formula. The other one is from my own. I have tested it for many different values and it seems to be effective. It assumes that the minimum value of a perimeter is more than the integer value of $\sqrt{4 \times triangles}$. If someone could once give me the mathematical demonstration of this I will be more confident in this use. These other peticuliar rules of my own are

not sure but usefull to fasten the process : the maximum values of a length digit that stop the research (for one shape number) : L[3]>10 when sides = 7 and L[3]>2 when sides = 8. I assume these rules are ok up to 100 triangles, but may be wrong for more than 100 triangles.
There are also some rules that are used when we ask for some symmetry. The different symmetries have been converted in rules for the digits, in order to fasten the program.

b) To find a solution

It has been already explain that we take each tan, starting with the biggest, and try to substract its DT digits to those of the shape, or remaining shape if some tans have alreaby be placed. This backtracking procedure has been adapted from the Eric's program. The idea is to explore all the possibilities for a tan to find a place inside the shape, until the shape is matched by the tans. When the program has been through all the possibilities of location (parameter i and j) and alias (8 alias maximum for each tan, see above), if the first tan has no more place to be, the shape is declared unmatchable by the tanset. If it is for the 5th tan to not find a place, we reward to the 4th tan and before increasing the location or the alias, we add back the DT digits that we have substract. If the 4th tan don't find its place, we reward to the 3d, ans so on. If it finds a new place, we substract the DT digit of the alias at its location and we try with the 5th tan...

c) To find all the solutions

Here it is easy, as we have recorded all the shapes in an arrayList. So we just have to try to match the different shapes, one by one, with the process that has been described in b).

d) To find a better tanset

Here also it is easy to go through all the process described in c) for any tanset. The procedure used here must find all the analogous tanset. This needs a special analysis to count the tans of each possible areas, and then to count the doubles for a special number of tan. When there are doubles in an area, we don't make any difference if the doubles are distributed in the same fashion as for the reference tanset. I mean if there is tans nb 2-2-3-3 there is 2 doubles for tans of area=2triangles. In this structure we will have the tansets composed of tans nb 2-2-3-3, 2-2-2-3, 2-3-3-3, and 2-2-4-4, 2-4-4-4, 2-2-2-4, 3-3-4-4, ...

For instance, the Tangram set has this structure :

```
TS[0][0]=1 TS[0][1]=2 TS[0][2]=1
TS[1][0]=2 TS[1][1]=3 TS[1][2]=0
TS[2][0]=4 TS[2][1]=2 TS[2][2]=1
T[0]=1 N[0]=2
T[1]=2 N[1]=1
T[2]=3 N[2]=1
T[3]=4 N[3]=1
T[4]=9 N[4]=2
```

The first line means that the first categorie of tans has an area of 1 triangle, there are 2 such tans and 1 double. The second line gives the tans of 2 triangles and the third the tans of 4 triangles. The other arrays T[] and N[] are for the number of the selected different tans. Here there are 5 different tans, which numbers are 1, 2, 3, 4 and 9. The N array counts how many tans of each number is needed.

When we go throught a structure, we keep the same TS numbers, but we change the T numbers.

e) To find all the possible structures

This function requires to go through all the combinations of choices for the known number of tans. For instance if we have 6 tans and 16 triangles, we have such kind of possibilities for the structure :

- Without double : 1-2-2-2-3-6 ; 1-2-2-2-4-5 ; 1-2-2-3-3-5 ; 1-2-2-3-4-4 ; 1-2-3-3-3-4 ; 2-2-2-3-3-4 ; 2-2-3-3-3-3 and that is all according there is only 3 tans of 2 triangles and 4 tans of 3 triangles.

- With 1 double : 1-1-2-2-2-8 1-1-2-2-3-7 1-1-2-2-4-6 1-1-2-2-5-5 1-1-2-3-3-6 1-1-2-3-4-5 1-1-2-4-4-4 1-1-3-3-3-5 1-1-3-3-4-4 1-2-2-2-2-7 ... and many others.
- With 2 doubles : 1-1-1-2-2-9 1-1-1-2-3-8 1-1-2-2-2-8 ... 1-1-2-2-2-8 1-1-2-2-3-7 1-1-2-2-4-6 1-1-2-2-5-5 1-1-2-3-3-6 ... 1-2-2-2-2-7 ... and many others.
- With up to 4 doubles : (1-1-1-1-1-11) ; 1-3-3-3-3-3 ; 2-2-2-2-2-6 ; 2-2-2-2-4-4 ; 2-2-3-3-3-3 and that is all.

This tanset 1-1-1-1-1-11 may be not intersting as a tanset but it shows another limit of the program in the present form : tan of 11 triangles area are very numerous (around 45 000 different shapes, including only 7 convex shapes) and are not in the pools of the program. In the present version we have put all the tans up to 6 triangles (107 different shapes of 6 triangles) but then we have limited the pool to the convex tans, and not all of them. This is due to a limitation parameter used to count different types of lengths and angles. The maximum length of a tan's side may not be more than 4 in the grid direction and 3 in the diagonal direction. This could be changed, but the need of tans exceeding 6 triangles area is not so important. If we deal with bigger tans we will have so much possibilities that the running time will be too big. So the real limitation of the program is there : too much treatments that take too much time. The code may be optimized, or wroten in C, a faster langage..

To explore the different types of structures that are possible. We keep the total number of triangles and the total number of tans. Of course this last number may be changed also, but here we assume that we keep it unchanged (it is easy to change it with another choice of tans in the 'Choice' mode). The structure is thus defined by the TS array. We don't keep here the number of doubles, or any number of the TS array. The first proposals are the structures without double, then with only 1 double, and so on. The proposals are in a combo box for the user information and choice. If one of those proposals is selected, the program displays the total amount of such tansets and when the 'validate' button is pressed, the program go on like in d).

To get a valid structure we just look all the numbers of n digits (n is the number of tans) and keep only those which a digit sum equal to the number of triangles and with rigth digits equal or greater than left digits. The digit 0 is used instead of 10 for tans of 10 triangles area. For instance, structure 1-2-2-2-3-6 is found in the number 122236 and structure 1-1-1-1-2-10 is found in the number 111120. For the doubles we just have to share them amongst the tans, first to the tans exceeding the natural number of tans of one type (there is only 1 tan of 1 triangle, so if we have a structure like 111120 we must have 3 doubles of this tan). When there are different possibilities to share the doubles, each possibilities are used : for instance, the structure 222334 may have 2 doubles of a 2 triangles tans and 1 of a 3 triangles tan, so if we are looking for tansets of 2 doubles, there could be 2 possibilities 222334 and 222334.