Correction bac-pratique-21052015-sc-16h-s3 (plus de suites)

Soit un nombre $n \in [10, 99]$
On souhaite générer toutes les suites de sommes d'entiers consécutifs de ***n***.

Solution naïve : brute force naïf

Pour trouver les suites de sommes consécutifs, on procède comme suit :
on commence par le début de la suite ***i*** (au début ***i*** = 1).

1) tant que la somme est inférieur strictement à ***n***, ajoute 2, puis 3, puis 4, ….
2) on vérifie, si somme est égale à ***n***, on sauvegarde le début de la suite et la fin de la suite qui est la valeur obtenu à l'étape (2)
3) on répète l'étape (1) et (2) en passant à 2 (***i***=2) et en ajoutant à partir du 3

exemple :
pour ***n*** = 12

i	**s**	
	0	
1	1	
	1+2	
	1+2+3	
	1+2+3+4	
	1+2+3+4+5 > 12 (on s'arrête)	
	0	
2	2 + 3	
	2 + 3 +4	
	2 + 3 + 4 + 5 > 12 (on s'arrête)	
	0	
3	3	
	3 + 4	
	3 + 4 + 5 = 12 (on s'arrête)	**d** **1** **2** **3** **4** **5** 3 **f** **1** **2** **3** **4** **5** 5
4,5,6	Même travail	**Pas de suites.**

Pour avoir ce résultat, il faut utiliser deux boucles pour générer les suites.

```
type
	tab = array[1..5] of byte;
```

```
procedure saisir (var a: byte ; msg: string);
begin
	repeat
		write(msg);
		readln(a);
	until (a > 9) and (a < 100);
end;
```

```
//Utilisation de deux boucles
//Complexité temporelle : O(n²)
procedure generSuites(n: byte ; var d,f: tab ; var nbs: byte);
var
	i, s, j: byte;
begin
	nbs := 0;
	for i := 1 to n div 2 do begin
		s := 0;
		j := i;
		while (s < n) do begin
			s := s + j;
			j := j + 1;
		end;
		if s = n then begin
			nbs := nbs + 1;
			d[nbs] := i;
			f[nbs] := j-1;
		end;
	end;
end;
```

```
procedure afficher(a: byte ; d,f: tab ; nbs: byte);
var
	i, j: byte;
begin

	for i:= 1 to nbs do begin
		write(a, ' = ');
		for j := d[i] to f[i] do
			write(j,'+');
		writeln;
	end;
end;
```

```
//MAIN
var
        n, m, nbsN, nbsM: byte;
        dN, fN, dM, fM: tab;
begin
        saisir(n, 'n= ');
        saisir(m, 'm= ');
        generSuites(n, dN, fN, nbsN);
        generSuites(m, dM, fM, nbsM);
        if  nbsN <> 0 then
                if nbsN > nbsM then afficher(n, dN, FN, nbsN)
                else if  nbsN < nbsM then afficher(M, dM, fM, nbsM)
                else begin
                        afficher(n, dN, fN, nbsN);
                        afficher(M, dM, fM, nbsM)
                end;
end.
```

Solution avec une seule boucle : brute force intelligent

Pour un nombre n :

on peut avoir n = 1 + 2 + 3 + 4 + ….+ y = y (y+1) / 2

aussi n = 4 + 5 + 6 + ...+ z = 1 + 2 + 3 + 4 + 5 + 6 + ...+ z – (1 + 2 + 3) = z (z+1) / 2 – 3 . 4 / 2

généralement :

$$n = y\frac{(y+1)}{2} - x\frac{(x+1)}{2}$$

$$n = \frac{y^2+y}{2} - \frac{x^2+x}{2} <==> 2n = y^2+y-x^2-x <==> y^2+y-(x^2+x+2n) = 0$$

avec x est le début de la suite -1 et y est la fin de la suite .

On va utiliser une seule boucle pour donner la valeur de x, donc il suffit de calculer la valeur de y.

On a un équation de second de degré à résoudre :

$$a = 1, b = 1, c = -(x^2+x+2n)$$
$$delta = 1+4(x^2+x+2n)$$

$$y1 = \frac{-1+\sqrt{1+4(x^2+x+2n)}}{2},\ y2 = \frac{-1-\sqrt{1+4(x^2+x+2n)}}{2}$$

$$y2 < 0, \text{à éliminer} ==> y = \frac{-1+\sqrt{1+4(x^2+x+2n)}}{2}$$

exemple :

$n=21, on\ cherche\ y\ pour\ tout\ les\ x, x\in[0,10]$

<table>
<tr><th>x</th><th>y</th><th></th></tr>
<tr><td>0</td><td>6.000000</td><td>
<table>
<tr><th colspan="5">d</th></tr>
<tr><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th></tr>
<tr><td>1</td><td></td><td></td><td></td><td></td></tr>
</table>
<table>
<tr><th colspan="5">f</th></tr>
<tr><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th></tr>
<tr><td>6</td><td></td><td></td><td></td><td></td></tr>
</table>
</td></tr>
<tr><td>1</td><td>6.152067</td><td>Non accepté</td></tr>
<tr><td>2</td><td>6.446222</td><td>Non accepté</td></tr>
<tr><td>3</td><td>6.865460</td><td>Non accepté</td></tr>
<tr><td>4</td><td>7.389867</td><td>Non accepté</td></tr>
<tr><td>5</td><td>8.000000</td><td>
<table>
<tr><th colspan="5">d</th></tr>
<tr><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th></tr>
<tr><td>1</td><td>6</td><td></td><td></td><td></td></tr>
</table>
<table>
<tr><th colspan="5">f</th></tr>
<tr><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th></tr>
<tr><td>6</td><td>8</td><td></td><td></td><td></td></tr>
</table>
</td></tr>
<tr><td>6</td><td>8.678780</td><td>Non accepté</td></tr>
<tr><td>7</td><td>9.412114</td><td>Non accepté</td></tr>
<tr><td>8</td><td>10.188779</td><td>Non accepté</td></tr>
<tr><td>9</td><td>11.000000</td><td>
<table>
<tr><th colspan="5">d</th></tr>
<tr><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th></tr>
<tr><td>1</td><td>6</td><td>10</td><td></td><td></td></tr>
</table>
<table>
<tr><th colspan="5">f</th></tr>
<tr><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th></tr>
<tr><td>6</td><td>8</td><td>11</td><td></td><td></td></tr>
</table>
</td></tr>
<tr><td>10</td><td>11.838963</td><td></td></tr>
</table>

La procédure "genererSuites" :

```
/Utilisation d’une seule  boucle
//Complexité temporelle : O(n)
procedure generSuites(n: byte ; var d,f: tab ; var nbs: byte);
var
        x: byte;
        y, delta: real;

begin
        nbs := 0;
        for x := 0 to n div 2  do begin
                delta := 1 + 4 * (sqr(x)+x+2*n);
                y := (-1+  sqrt(delta)) / 2;
                 if frac(y) = 0.0 then begin
                        nbs := nbs + 1;
                        d[nbs] := x + 1;
                        f[nbs] := trunc(y);
                 end;
        end;
end;
```