Nº matrícula:_____ **Nombre:** _____

Apellidos: _____

Módulo 4. Programación dinámica.

Diseña e implementa el método *public boolean haySuma(int[] valores, int v)* en Java basado en el esquema de programación dinámica, que permita determinar si, dado un conjunto de números enteros positivos mayores que 0, existe un subconjunto que sume el valor *V*. Por ejemplo, dado el vector *valores = {5, 8, 7, 1, 3, 1}* y el valor *v=10*, el algoritmo debería devolver *true*; dado el vector *valores = {5, 8, 7, 2, 3}* y el valor *v=21*, el algoritmo debería devolver *false*.

a) Define la **entrada**, la **salida** y la **semántica** de la función sobre la que estará basado el algoritmo de programación dinámica.

HaySuma(i,j): es posible conseguir valores con un acumulado total de i usando los elementos 0..j del vector valores.
Entrada: i>=0, j>=0 (2 enteros). El valor j=0 indica que se dispone del elemento de la posición 0 del vector valores. El valor i=0 indica un acumulado total de 0.
Salida: valor booleano (indica si existe el subconjunto)

b) Expresa recursivamente la función anterior.

```
j>=0 -> HaySuma(0,j) = true
i>0   -> HaySuma(i,0) = true si valores[0]==i y  false en caso contrario

i>0 y j>0 HaySuma(i,j) = true si HaySuma(i, j-1)=true
                       = true si HaySuma(i-valores[j], j-1) = true
                       = false en caso contrario
```

c) Basándote en los anteriores apartados implementa el algoritmo en Java siguiendo el esquema de *Programación Dinámica*.

```java
public boolean haySuma(int[] valores, int v){
   boolean[][] hay = new boolean[v+1][valores.length];
   hay[0][0] = true;
   for (int i=1; i<=v; i++) hay[i][0]= (valores[0]==i);

   for (int j=1; j<valores.length; j++)
      for (int i=0; i<=v; i++)
         if (hay[i][j-1]) hay[i][j] = true;
         else if ((valores[j] <=i) && (hay[i-valores[j]][j-1]))
            hay[i][j]=true;
         else hay[i][j]=false;
   return hay[v][valores.length-1];
}
```

```java
/* La versión con memoria optimizada sería */

public boolean haySumaOptimizado(int[] valores, int v){
   boolean[][] hay = new boolean[v+1][2];
   hay[0][0] = true;
   for (int i=1; i<=v; i++) hay[i][0]= (valores[0]==i);

   for (int j=1; j<valores.length; j++)
      for (int i=0; i<=v; i++)
         if (hay[i][(j-1)%2]) hay[i][j%2] = true;
         else if ((valores[j] <=i) && (hay[i-valores[j]][(j-1)%2]))
            hay[i][j%2]=true;
         else hay[i][j%2]=false;
   return hay[v][(valores.length-1)%2];

}
```