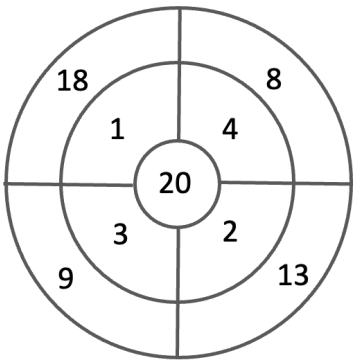
Nº matrícula:_____ **Nombre:** _____

Apellidos: _____

Problema. Programación dinámica.



Batman y Wonder Woman son grandes aficionados al juego de los dardos. El tablero con el que habitualmente juegan está dividido en secciones, cada una de ellas con un valor numérico concreto, tal y como se muestra en la figura.

Wonder Woman le plantea a Batman el reto de conseguir una puntuación concreta P, usando el menor número de dardos. Dado que a Batman no se le dan bien los problemas de optimización, pide ayuda a los alumnos de Algorítmica de la ETSISI para que le diseñen e implementen un algoritmo con la siguiente cabecera ***int[] numDardos(int[] secciones, int P)***, que le indique a qué secciones del tablero debe tirar los dardos para conseguir obtener la puntuación P minimizando el número de dardos usados (considerar que Batman es un gran jugador de dardos y todos los dardos que tira quedan situados en alguna sección del tablero puntuable).

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

minDardos(i,j): número mínimo de dardos que debe tirar Batman cuando dispone de las secciones 1..i y la puntuación a conseguir es j
Entrada: i>=0, j>=0 (2 enteros). El valor i=0 indica que no hay secciones a las que tirar dardos. El valor j=0 indica que la puntuación a conseguir es 0.
Salida: valor entero (número mínimo de dardos)

b) Expresa recursivamente la función anterior.

i>=0 -> minDardos(i,0)= 0 (el valor a conseguir es 0)
j>0 -> minDardos(0,j) = ∞ (no se dispone de secciones y el valor a conseguir >0)

i>0 y j>0 minDardos = min(minDardos(i-1,j), 1+minDardos(i, j-secciones[i])

NOTA: con *secciones[i]* hacemos referencia a la zona del tablero i, teniendo en cuenta que las secciones se numeran 1..N.

c) Basándote en los anteriores apartados implementa un algoritmo en Java, basado en un esquema de programación dinámica, que dada una puntuación concreta P, devuelva cuántos dardos debe tirar Batman a cada una de las secciones del tablero, teniendo en cuenta que el número de dardos a tirar debe ser el mínimo posible.

```
int[] numDardos(int[] secciones, int P){
    int[][] cant = new int[2][P+1];
    boolean[][] aux = new boolean[secciones.length+1][P+1];
    for (int c=0;c<=P; c++) {
        cant[0][c] = Integer.MAX_VALUE; aux[0][c]= false;
    }
    cant[0][0] = 0; cant[1][0] = 0;
    for (int i=0;i<= secciones.length; i++)
        aux[i][0] = false;
    for (int i=1; i<= secciones.length; i++) {
        for (int c=1; c<=P; c++)
            if (c< secciones [i-1]) {
                cant[i%2][c]=cant[(i-1)%2][c];
                aux[i][c]=false;}
            else {
                cant[i%2][c]=Math.min(cant[(i-1)%2][c],
                                      1+cant[i%2][c- secciones [i-1]]);
                aux[i][c]=(cant[i%2][c]==1+cant[i%2][c- secciones [i-1]]);}
    }
    return cantidadDardos(aux, secciones);
}

int[] cantidadDardos(boolean[][] aux, int[] secciones){
    int i=aux.length-1;     int c=aux[0].length-1;
    int[] decision = new int[aux.length-1];
    decision[i-1]=0;
    while (c>0 && i>0) {
        if (aux[i][c]==true)  {
            c=c- secciones [i-1]; decision[i-1]++; }
        else {
            i--; decision[i-1]=0;}
    }
    return decision;
}
```