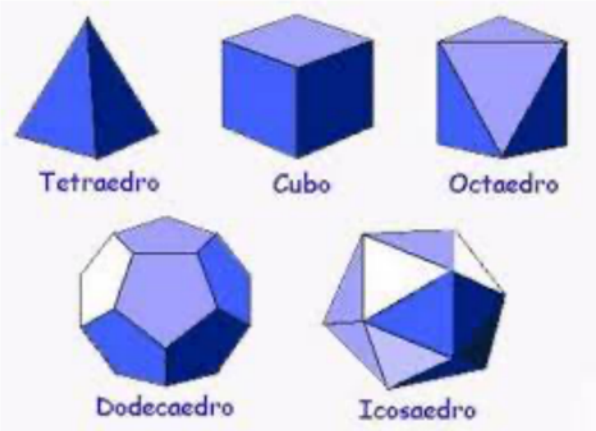
Examen de Algorítmica y Complejidad (Plan 2014)

30 de junio de 2023

Nº matrícula:_____ **Nombre:** _____

Apellidos: _____

Problema. Se quiere formar una **torre** de objetos con los **cinco sólidos platónicos**(1) (tetraedros, hexaedros (cubos), octaedros, dodecaedros e icosaedros) que en total sumarán N caras. Por ejemplo, una torre formada por tres solidos: [1 icosaedro, 1 dodecaedro, 1 tetraedro] sumaría un total de 36 caras. Para formar una torre válida, la base y los elementos intermedios deben poder sostenerse sobre una cara superior y otra inferior, es decir un sólido no puede apoyarse sobre una arista. Es por ello que en una torre válida el tetraedro solo podrá estar en la cima y sólo podrá haber 1 como máximo.



(1) Un tetraedro tiene 4 caras, un hexaedro (cubo) tiene 6 caras, un octaedro tiene 8, un dodecaedro tiene 12 y un icosaedro 20.

Aclaración: La torre no tiene por qué ser estable desde un punto de vista físico.

SE PIDE: Implementar un algoritmo basado en el **esquema Voraz** que dado un numero v de caras, devuelva cuántos objetos de cada tipo han sido necesarios. En caso de que el problema no tenga solución se devolverá ***null***. El algoritmo deberá tener la siguiente cabecera:

```
int[] greedyTower(int v)
```

donde ***v*** es el número exacto de caras que intentamos obtener en la torre que queremos construir, e ***int[]*** es un vector de enteros que representa el número de [tetraedros, cubos, octaedros, dodecaedros, icosaedros] que se han empleado para construir la torre

Ejemplos de posibles soluciones:

- para v= 168: solución -> [0, 0, 1, 0, 8]
- para v= 38: solución -> [0, 1, 0, 1, 1]
- para v= 25 solución -> null

Aclaraciones: Se podrán implementar todos los métodos adicionales que se consideren necesarios.

```java
int[] greedyTower(int v){
    //solucion[0]->tetraedro; solucion[1]->hexaedro (cubo);
    //solucion[2]->octaedro; solucion[3]->dodecaedro;
    //solucion[4]->icosaedroint[]
    int[] solucion = new int[5];
    for (int i=0; i<solucion.length; i++) solucion[i]=0;

    int faltanCaras = v, i=0, c;
    int[] candidatos = {20, 12, 8, 6};

    while(i < candidatos.length && faltanCaras>0){
        c= faltanCaras/candidatos[i];
        if (c>0){
            faltanCaras = faltanCaras%candidatos[i];
            solucion[solucion.length-i-1]=c;
        }
        i++;
    }
    if(faltanCaras==4) {
        solucion[0]=1;
        faltanCaras=0;
    }
    if(faltanCaras==0)
        return solucion;
    else
        return null;

}
```