CC1001-Computación I – Control Nº 1 – lunes 9 de abril de 2007 - Tiempo: 1 hr 45 minutos
Con apuntes individuales – Sin consultas - Contestar en hojas separadas
Importante. No necesita incluir la parte inicial de los programas (import, class, main).

Pregunta 1

a) (2 puntos) Escriba un método (función) de encabezamiento **static public int sumaDivisores(int x)** que entregue la suma de los divisores propios del número x (incluye 1 pero excluye x).

```
int suma=0;
//control de ciclo: 0.5 puntos
for(int i=1; i<x; ++i)
  //sumar divisores propios y devolverlo: 1.5 ptos
  if( x%i==0)
    suma=suma+i;
return suma;
```

b) (4 puntos) Dos números amigos son dos enteros positivos distintos tales que la suma de los divisores propios de uno de ellos es igual al otro. Los primeros dos números amigos son 220 y 284, ya que los divisores propios de 220 son 1, 2, 4, 5, 10, 11, 20, 22, 44, 55 y 110, que suman 284 y los divisores propios de 284 son 1, 2, 4, 71 y 142, que suman 220.

Escriba un programa que, utilizando la función anterior, encuentre los siguientes dos números amigos (que son menores que 2.147.483.647, que es el nº máximo de tipo int).

Solución 1:

```
//control de ciclos: 1 punto
int i, j;
for(i=285; i<maxInt; ++i)
{
  for(j=i+1; j<maxInt; ++j)
  {
    //detectar números amigos: 1.5
    if(sumaDivisores(i)==j
    && sumaDivisores(j)==i
    && i!=j)  //detectar números perfectos (amigos de sí mismos)
    //quebrar los dos ciclos: 1.0
      break;
  }
  if(j<maxInt) break;
}
//mostrar números amigos: 0.5
U.println(i+" "+j);
```

Solución 2 (mucho más eficiente):

```
//repetir para números desde 285: 0.5
for(int i=285; true; ++i){
  //sumar divisores de número y de su suma: 1.5 puntos
  int j=sumaDivisores(i);
  int s=sumaDivisores(j);
  //detectar números amigos: 1.0
  if(s==i
  && j!=i){
   //mostrar amigos y terminar: 1.0
    U.println(i+" "+j);
    break;
  }
}
```

Pregunta 2

a)(1 punto) Escriba una función que se invoque en la forma distancia(x,y,z) y que entregue un nº real que representa la distancia entre el origen de coordenadas (0,0,0) y el punto en el espacio de coordenadas reales (x,y,z).

```
static public double distancia(double x,double y,double x){//0.3
  return Math.sqrt(x*x + y*y + z*z); //0.7
}
```

b)(5 puntos) Escriba un programa que, utilizando la función anterior, determine el punto más alto y el punto más lejano que alcanza un proyectil que fue lanzado al espacio en el punto de coordenadas (0,0,0). El programa debe establecer el diálogo siguiente:

```
tiempo 1:
…
tiempo n:
x?
y?
z? 0 (fin: volvió a tierra)
distancia=n°
punto más alto= n° n° n° tiempo=n°
punto más lejano= n° n° n° tiempo=n°
```

Notas.

- En cada unidad de tiempo el programa debe solicitar las coordenadas reales del proyectil y mostrar la distancia desde el origen (0,0,0)
- Los puntos más alto y más lejano deben mostrarse con sus coordenadas x, y, z y el tiempo en que se alcanzaron (un nº entero entre 1 y n)

```
//inicializar variables globales: 0.7
double dmax=0, x1, y1, z1; int t1;
double z2=0, x2, y2; int t2;
//repetir incrementando tiempo: 0.7
for(int t=1; true; ++t)
{
 //leer coordenadas: 0.3
 double x=U.readDouble("x?"),y=U.readDouble("y?"),z=U.readDouble("z?");
 //calcular y mostrar distancia: 0.5
 double d=distancia(x,y,z);
 U.println("distancia="+d);
 //mantener mayor distancia: 0.8
 if(d>dmax){
   dmax=d; x1=x; y1=y; z1=z; t1=t;
 }
 //mantener mayor altura: 0.7
 if(z>z2){
   z2=z; x2=x; y2=y; t2=t;
 }
 //terminar al llegar a tierra: 0.5
 if(z==0) break;
}
//mostrar resultados: 0.8
U.println("punto más alto="+x2+" "+y2+" "+z2+" tiempo="+t2);
U.println("punto más lejano="+x1+" "+y1+" "+z1+" tiempo="+t1);
```