Asignatura: Informática.
Titulación: Licenciado en Matemáticas.
Convocatoria: 31/02/2006.
Parte I: Conceptos y técnicas básicas.

Cada ejercicio puntúa un máximo de 0.6 puntos. La calificación máxima en esta parte es de 3 puntos.

Ejercicio 1. Sean a, b, c variables de tipo booleano, x, y, z variables de tipo entero. Determinar, suponiendo que utilizamos la sintaxis del peudocódigo, cuáles de las expresiones siguientes son sintácticamente incorrectas justificando la respuesta.

(a) $(3 * z \geq 100) \vee (\neg a \wedge b)$
(b) $(a \vee c) = \neg(y \geq z)$
(c) $(x > a) \wedge (y \; div \; 5 < 3)$
(d) $z - 2 < y + 3$

Ejercicio 2. Considérese el siguiente fragmento de código:

```
x, y, z: entero;
{P}
x:=x+z;
y:=z-x;
z:= z+x;
{x=X, y=Y, z=Z}
```

dónde P es un predicado y X, Y, Z, son variables de especificación. Determinar los predicados P que hacen correcta la especificación anterior.

Ejercicio 3. Considerar la siguiente acción.

```
acción tratarVector(e/s t:tabla[1..N] de entero  ent i: entero)
{Pre.- N>=1, t=T}
        si (i<= N) entonces t[i]:=t[i]+t[i-1]; tratarVector(t,i+1); fsi
{Post.- ???}
facción
```

Indicar, explicando razonamente la respuesta, cuál será el valor de cada componente t[i] para cada i, 1<=i <=N, en función de los valores iniciales de la tabla de entrada: T[1], T[2],…, T[N] tras la llamada tratarVector(t,2).

Nota. Distinguir el caso, N=1 del caso N>1.

Ejercicio 4. Especificar una función , tal que dada una tabla de números enteros $a[1...N]$ y un número entero $k > 0$, calcule el número de parejas (i, j) tales que $1 <= i < j <= N$ y $a[j] - a[i] >= k$.

A continuación se pide diseñar un algoritmo iterativo para resolver la especificación anterior siguiendo el esquema de recorrido de tablas.

Nota: el bucle (o bucles) utilizados en el diseño deben ir acompañados de su correspondiente predicado invariante. Se valorará la adecuada construcción y expresión clara del invariante y el ajuste del diseño al invariante propuesto.

Ejercicio 5. Considérese la siguiente especificación

S: secuencia de entero;
ss: entero;
encontrado: booleano;
{**Pre**.- S es una secuencia no vacía, pi(S)= vacío, pd(S)=S, ss contiene la suma de todos los elementos de la secuencia S}

IgualSumaPosteriores

{**Post**.- encontrado indica si existe algún elemento en S igual a la suma de todas los elementos posteriores a dicho elemento en la secuencia,
encontrado-> S.actual() igual a la suma de los elementos posteriores a él en la secuencia y ningún elemento anterior satisface dicha propiedad.
no encontrado-> nos encontramos en situación de fin de secuencia.}

Se pide, diseñar un algoritmo que satisfaga la especificación anterior y que se ajuste al siguiente predicado invariante:

{I= "ningún elemento anterior al actual satisface la propiedad, ss contiene la suma de todos los elementos posteriores al actual incluído éste,
encontrado-> S.actual() igual a la suma de los elementos posteriores a él en la secuencia."
}

Examen de Informática (Licenciado en Matemáticas)

Primer examen. Febrero 2006

Parte II. Problema (3 puntos=50% nota del examen)

Se dispone de un programa capaz de obtener soluciones al problema de optimización de una función de dos variables $f(x,y)$. Este programa deposita las soluciones (compuestas por tres valores: x, y, $f(x,y)$) en un objeto de la clase `Solucion`, ya realizada, cuya interfaz se muestra a continuación, y que representa una secuencia:

```
public class Solucion {

  /** crea una secuencia */
  public Solucion() {...}

  /** reinicia la secuencia haciendo que la solucion actual sea
   *  la primera de la secuencia*/
  public void reinicia() {...}

  /** avanza a la siguiente solucion de la secuencia*/
  public void siguiente() {...}

  /** Obtiene el valor de la funcion f(x,y) de la solucion actual*/
  public double valorActualF() {...}

  /** Obtiene el valor de la variable x de la solucion actual*/
  public double valorActualX() {...}

  /** Obtiene el valor de la variable y de la solucion actual*/
  public double valorActualY() {...}

  /** Indica si se ha alcanzado o no el final de la secuencia */
  public boolean finDeSecuencia() {...}

  /** borra la secuencia dejandola con cero soluciones */
  public void borra() {...}

  /** inserta una solucion en la secuencia, en ultimo lugar */
  public void inserta(double valorF, double x, double y) {...}

  /** retorna la media de los valores f(x,y) de todas las soluciones
   * contenidas en la secuencia */
  public double media () {...}

}
```

Se pide hacer otra clase con operaciones para explotar los resultados almacenados en un objeto de esa clase `Solucion`. La clase a realizar responde al siguiente diagrama de clases:

Resultados
Solucion sol
Resultados (Solucion soluc) Solucion **filtra**(double porcentajeMinimo) boolean **haySolucionEnZona** (double xMin, double xMax, double yMin, double yMax) double **mejorSolucionZona** (double xMin, double xMax, double yMin, double yMax)

La función a realizar por cada una de las operaciones de esta clase es la siguiente:

- Constructor: copia la referencia `soluc` (que se refiere a un objeto de la clase `Solucion`) en el atributo `sol`
- `filtra`(): retorna un objeto de la clase `Solucion` que contiene otra secuencia con todas las soluciones almacenadas en la secuencia `sol` cuyo valor *f*(*x*,*y*) es superior a la media de las soluciones de `sol` multiplicada por `porcentajeMinimo`/100
- `haySolucionEnZona`(): retorna un booleano que indica si existe o no una solucion al menos, en la zona cuyos valores *x* están entre [`xMin`,`xMax`] y cuyos valores *y* están entre [`yMin`,`yMax`]
- `mejorSolucionZona`(): retorna el máximo de las soluciones *f*(*x*,*y*) almacenadas en `sol` y en la zona cuyos valores *x* están entre [`xMin`,`xMax`] y cuyos valores *y* están entre [`yMin`,`yMax`]. Si no hay ningún valor en la zona retorna el mínimo valor real, que en Java es `-Double.MAX_VALUE`

Usar para `filtra`() y `mejorSolucionZona`() el esquema de recorrido en secuencias, y para `haySolucionEnZona`() el esquema de búsqueda en secuencias.

Los tres métodos de la clase `Resultados` se valorarán cada uno en un 30% de la nota del problema. El constructor y el resto de la clase en un 10%.