Asignatura: Programación I.
Titulación: Ingeniería Informática.
Convocatoria: 10/02/2006.
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. Un número natural n es creciente si cada dígito de n es menor o igual que el que está a su derecha. Por ejemplo, 2349 es creciente mientras que 3492 no lo es.

Un número natural con un solo dígito –es decir, menor o igual que nueve- se considera creciente. En el caso de tener más de un dígito será creciente si lo es n **div** 10 (el natural formado por los dígitos más altos de n) y además el último dígito de n, n **mod** 10, es mayor o igual que el último dígito de n **div** 10 – es decir, el número dado por (n **div** 10) **mod** 10.

Para ilustrar la discusión anterior notar que si n=2349, n **div** 10=234 que es creciente, (n **div** 10) **mod** 10= 4 que es menor o igual que n **mod** 10 =9.

Se pide diseñar una función recursiva que determine si un número natural es creciente:

```
función creciente (n: entero) retorna c: booleano
{Pre. n>=0}

{Post. c indica si n es creciente}
función
```

Indicar en el diseño cuál es el caso directo y cuál el recursivo.

Ejercicio 2. Considérese la siguiente especificación que calcula cierta noción de distancia entre dos tablas de enteros.

```
función distancia (A, B: tabla[1..N] de entero) retorna i: entero
{Pre.- N>1, A está ordenada decrecientemente, B está ordenada crecientemente,
A[1]>=B[1], A[N]<B[N] }

{Post.- A[i]>=B[i], A[i+1]<B[i+1]}
ffunción
```

Por ejemplo para las siguientes tablas la función haría retornar la posición i=5 indicada en la figura.

A:	15	13	9	8	6	3	1	-2	-7	-9
B:	-4	-1	1	2	3	4	7	10	12	13

Utilizando una estrategia de búsqueda iterativa sobre tablas indicar cuál es la propiedad buscada y proponer un invariante. Finalmente diseñar la función completa ajustándose al esquema y al invariante.

Ejercicio 3. Sean a, b, c variables de tipo booleano; x, y, z, variables de tipo real. Si usamos la notación del pseudocódigo indicar cuáles de las siguientes asignaciones, justificando la razón, son incorrectas:

(a) $a := (x >= 1.0) \wedge (z <= x)$
(b) $x := (y \ div \ x) + 3.1 * z$
(c) $a := (\neg b \vee c) \wedge (x - y \neq 5.0)$
(d) $c := (x = y) \wedge (z / 2.3 < x)$

Ejercicio 4. Se consideran dos secuencias S1 y S2 de elementos de un cierto tipo de datos para el cual se dispone de un operador de comparación (=) de resultado booleano. Sobre las secuencias S1, S2 y T se pueden aplicar las operaciones conocidas de la intefaz de la clase secuencia: **actual**(), **avanzar**(), **fds**(), **preparar**(), **escribir**(), disponiéndose además de un método **reiniciar**() que hace que el elemento actual sea el primero de la secuencia.

Se pide especificar y diseñar un algoritmo que genere otra secuencia T formada por los elementos comunes a S1 y a S2 siguiendo el esquema que se detalla a continuación:

```
inicializaciones;
{Inv: T contiene los elementos comunes a pi(S1) y a S2 }
mientras  no S1.fds() hacer

        inicializaciones;
        {Inv: el elemento actual de S1 no ha sido encontrado en pi(S2)}

        mientras no S2.fds() y no encontrado hacer
                actualizar encontrado;
                tratamiento en caso de no encontrado;
        fmientras

        {encontrado indica si el elemento actual se S1 está en S2,
         encontrado-> actual de S1 coincide con actual de S2,
        no encontrado-> S2 está en fin de secuencia}

        añadir actual de S1 a T si encontrado;
        avanzar en S1;

fmientras
```

Nota: No se admitirán soluciones "imaginativas" que no se ajusten al esquema indicado anteriormente.

Ejercicio 5. Dado el siguiente fragmento de código:

```
p, q: entero;
{Pre.- q es par ó bien p es impar}
si
        q mod 2= 1 -> q:=q-1
        q mod 2= 0 -> q:=2*p
fsi;
p:=p+1;
{Post.- ??}
```

Se pide determinar razonadamente cuáles de las siguientes postcondiciones hacen correcta la especificación anterior:

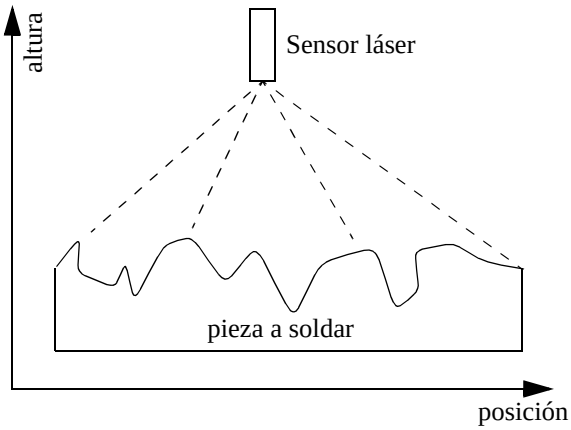
(a) q es par ó bien p es impar.
(b) q es par y p es impar.
(c) p es impar.

Examen de Programación I (Ingeniería Informática)

Febrero 2006

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

Una máquina de soldadura dispone de un sensor láser capaz de determinar un perfil de las altitudes de la pieza que está siendo soldada, tal como se muestra en la figura. Se desea hacer software que sirva para que la máquina almacene los perfiles y obtenga información sobre ellos.



Se dispone de una clase ya realizada, llamada `Punto`, que permite almacenar un dato de un perfil de altura. El dato está compuesto por una altura y una posición, ambos en milímetros. La interfaz de esta clase se muestra a continuación.

```
public class Punto {

  /** Constructor al que se le pasa la altura y posición del punto */
  public Punto(double altura, double posicion) {...}

  /* Retorna la altura del punto */
  public double altura() {...}

  /* Retorna la posición del punto */
  public double posicion() {...}
}
```

Se dispone de otra clase, llamada `Perfil`, para almacenar una tabla de objetos de la clase `Punto` en la que se guardan los datos de un perfil de elevación completo obtenido mediante el sensor láser. La tabla se numera con índices que comienzan en cero, y tiene un tamaño variable pero limitado. La clase está parcialmente implementada, de la forma siguiente:

```
public class Perfil {
  /** Atributos privados: tabla de puntos y número actual de elementos
   */
  private Punto tabla[];
  private int num;

  /** Constructor, al que se le pasa el tamaño máximo de la tabla de
   *  puntos. Crea la tabla de ese tamaño y pone el número de elementos
   *  a cero
   */
  public Perfil(int tamanoMaximo) {
    tabla=new Punto[tamanoMaximo];
    num=0;
  }

  // métodos a escribir:
  public void inserta(double altura, double posicion) {...}

  public double alturaMedia() {...}

  public Perfil filtra() {...}

  public int primerPico() {...}
}
```

Lo que se pide es escribir los métodos cuyo contenido está sin completar:

- `inserta`()[1]: Si la casilla `num` de la tabla existe (es decir, si el número de elementos de la tabla, `num`, es menor que el tamaño de la tabla), crea un objeto de la clase `Punto` con la altura y posición indicados en los parámetros, copia la referencia a ese objeto en la casilla `num` de la tabla, e incrementa `num` en una unidad. Si la casilla no existe (porque el número actual de elementos es igual al tamaño de la tabla), el método no hace nada.
- `alturaMedia`(): Retorna la media ponderada de las alturas del perfil almacenadas en la tabla. La media se calcula como:

$$\frac{\sum_{i=0}^{num-2}\left(\frac{(h_{i+1}+h_i)}{2}\cdot(x_{i+1}-x_i)\right)}{x_{num-1}-x_0}$$

  siendo h_i la altura del punto *i*-ésimo y x_i la posición del punto *i*-ésimo de la tabla.

- `filtra`(): Crea y retorna un nuevo objeto de la clase `Perfil` que contiene los puntos contenidos en la tabla del objeto actual cuyo valor de altura está comprendido entre el 80% y el 120% de la altura media obtenida con `alturaMedia`(). Para ello, primero crea el nuevo objeto con el mismo tamaño máximo que la tabla. Luego recorre todos los puntos de la tabla, desde el `0` hasta el `num-1`, y para aquellos que cumplen el criterio indicado, los inserta con `inserta`() en la nueva tabla. Finalmente retorna el nuevo objeto. Seguir el esquema de recorrido iterativo en tablas.
- `primerPico`(): Retorna el índice del primer pico del perfil. En caso de que no haya ningún pico, retorna `-1`. Un pico se define como un punto del perfil tal que su valor de altura es mayor que las alturas tanto de su antecesor en la tabla como de su sucesor. Observar que con esta definición ni el primer punto ni el último de la tabla pueden ser picos. Seguir el esquema de búsqueda iterativa en tablas, adaptado a esta circunstancia.

La valoración de cada método es según su dificultad:

- inserta: 20%
- alturaMedia: 25%
- filtra: 30%
- primerPico: 25%

1. *Nota*: Las llamadas que se hagan a `inserta`() deben ser con puntos de `posición` creciente, para que queden ordenados por posición en la tabla. Este orden de las llamadas es responsabilidad de otra parte del programa, por lo que no hay que tenerlo en cuenta aquí.