Escuela de Ingeniería y Arquitectura - Depto. de Informática e Ingeniería de Sistemas
Examen práctico de Programación 2 - 15 de septiembre de 2016 - Tiempo hasta las **14:00**

Este examen práctico individual en laboratorio forma parte de la evaluación de la asignatura. Su calificación tiene un peso del 30 % en la calificación final de la asignatura en la convocatoria de septiembre.

En este examen práctico se han de diseñar tres funciones genéricas `C++` que trabajan con secuencias de datos definidas a partir del tipo genérico **`Secuencia`**. Las tres funciones genéricas a diseñar se han especificado en el listado presentado en la página posterior. Al proceder a su diseño se han de tener en cuenta las siguientes indicaciones.

1. El diseño de la función **`primera`**`(S,dato)` no podrá presentar ningún bucle. Cada una de las funciones que integren la solución deberá presentar un diseño correcto (especificación + código). Se valorará especialmente la eficiencia del algoritmo programado. [3.0 puntos]
2. El diseño de la función **`segunda`**`(S,S1)` no podrá presentar ningún bucle. Cada una de las funciones que integren la solución deberá presentar un diseño correcto (especificación + código). [3.0 puntos]
3. El diseño de la función **`tercera`**`(S,minimo)` no podrá presentar ningún bucle. Cada una de las funciones que integren la solución deberá presentar un diseño correcto (especificación + código). Se valorará especialmente la eficiencia del algoritmo programado.[4.0 puntos]

En la carpeta **`examenSeptiembre`**, accesible desde la web de la asignatura (en zona de materiales docentes comunes seleccionar *`Código C++ descargable`*), se encuentran dos ficheros para facilitar este trabajo:

- Fichero **`funcionesSecuencia.h`** con un listado análogo al presentado anteriormente.
- Fichero **`pruebas.cc`** con un programa que permite hacer algunas pruebas sobre el comportamiento de las tres funciones pedidas.

PRESENTACIÓN DEL TRABAJO

Como resultado del trabajo se entregará un único fichero cuyas primeras líneas sean un **comentario con el nombre del alumno** y, a continuación, que almacene exclusivamente el código de las funciones **`primera`**`(S,dato)`, **`segunda`**`(S,S1)` y **`tercera`**`(S,minimo)`, precedidas, en su caso, por sus funciones auxiliares.

La entrega se hará a través de la plataforma ***Moodle2*** (moodle2.unizar.es) antes de las 14:00. Se recomienda no hacerlo hasta que haya sido verificado de forma exhaustiva que el comportamiento de todo código diseñado es correcto.

```
// Código almacenado en el fichero funcionesSecuencias.h

// Número máximo de elementos que puede llegar atener una secuencia
const int MAX = 250;        // Redefinir su valor en caso necesario

/*
 * Un dato definido a partir del tipo genérico Secuencia representa una secuencia
 * de datos de tipo T
 */
template <typename T>
struct Secuencia {
    // El valor de S.n define el número de elementos de una secuencia
    // S = [e_1, e_2, ..., e_n] con S.n >= 0 y S.n <= MAX
    int n;
    // Las 'n' elementos de una secuencia S = [e_1, e_2, ..., e_n] son datos de tipo T
    // que se almacenan del siguiente modo en la tabla 'data':
    //      e_1 se almacena en S.data[0]
    //      e_2 se almacena en S.data[1]
    //       ...
    //      y, finalmente, e_n se almacena en S.data[v.n-1]
    T data[MAX];
};

/*******    Funciones a diseñar sin bucles en este examen práctico      *******/

/*
 * Pre: S.n >= 0 AND S.n <= MAX AND
 *       (PT alfa EN [0,S.n-2]. S.data[alfa] <= S.data[alfa+1])
 * Post: primera(S,dato) = I AND
 *         ((PT alfa EN [0,S.n-1]. S.data[alfa] != dato) -> I < 0 ) AND
 *         ((EX alfa EN [0,S.n-1]. S.data[alfa] = dato)  -> S.data[I-1] = dato)
 */
template <typename T>
int primera (const Secuencia<T> S, const T dato);

/*
 * Pre: S.n >= 0 AND S = So AND S1.n >= 0 AND S.n + S1.n <= MAX
 * Post: S.n = So.n + S1.n AND
 *         (PT alfa [0,So.n-1]. S.data[alfa] = So.data[alfa]) AND
 *         (PT alfa [0,S1.n-1]. S.data[So.n+alfa] = S1.data[alfa])
 */
template <typename T>
void segunda (Secuencia<T>& S, const Secuencia<T> S1);

/*
 * Pre: (PT alfa EN [0,S.n-2]. S.data[alfa] <= S.data[alfa+1])
 * Post: tercera(S,minimo) = (NUM alfa EN [0,S.n-1]. S.data[alfa] >= minimo)
 */
template <typename T>
int tercera (const Secuencia<T> S, const T minimo);
```

Una solución de las tareas propuestas en este examen

```
/*
 * Pre: S.n >= 0 AND S.n <= MAX AND desde >= 0 AND
 *      (PT alfa EN [desde,S.n-2]. S.data[alfa] <= S.data[alfa+1])
 * Post: primera(S,dato) = I AND
 *       ((PT alfa EN [desde,S.n-1]. S.data[alfa] != dato) -> I < 0 ) AND
 *       ((EX alfa EN [desde,S.n-1]. S.data[alfa] = dato)  -> S.data[I-1] = dato)
 */
template <typename T>
int primera (const Secuencia<T> S, const T dato, const int desde) {
    const int NO_ENCONTRADO = -1;
    if (desde < S.n) {
        if (S.data[desde] == dato) {
            return desde + 1;
        }
        else if (S.data[desde] < dato){
            return primera(S, dato, desde + 1);
        }
        else {
            return NO_ENCONTRADO;
        }
    }
    else {
        return NO_ENCONTRADO;
    }
}

/*
 * Pre: S.n >= 0 AND S.n <= MAX AND
 *      (PT alfa EN [0,S.n-2]. S.data[alfa] <= S.data[alfa+1])
 * Post: primera(S,dato) = I AND
 *       ((PT alfa EN [0,S.n-1]. S.data[alfa] != dato) -> I < 0 ) AND
 *       ((EX alfa EN [0,S.n-1]. S.data[alfa] = dato)  -> S.data[I-1] = dato)
 */
template <typename T>
int primera (const Secuencia<T> S, const T dato) {
    return primera(S, dato, 0);
}

/*
 * Pre: S.n >= 0 AND S = So AND S1.n >= 0 AND desde >= 0 AND S.n + S1.n - desde <= MAX
 * Post: S.n = So.n + S1.n - desde AND
 *       (PT alfa [0,So.n-1]. S.data[alfa] = So.data[alfa]) AND
 *       (PT alfa [desde,S1.n-1]. S.data[So.n+alfa-desde] = S1.data[alfa])
 */
template <typename T>
void segunda (Secuencia<T>& S, const Secuencia<T> S1, const int desde) {
    if (desde < S1.n) {
        S.data[S.n] = S1.data[desde];
        S.n = S.n + 1;
        segunda(S, S1, desde + 1);
    }
}
```

```
/*
 * Pre: S.n >= 0 AND S = So AND S1.n >= 0 AND S.n + S1.n <= MAX
 * Post: S.n = So.n + S1.n AND
 *       (PT alfa [0,So.n-1]. S.data[alfa] = So.data[alfa]) AND
 *       (PT alfa [0,S1.n-1]. S.data[So.n+alfa] = S1.data[alfa])
 */
template <typename T>
void segunda (Secuencia<T>& S, const Secuencia<T> S1) {
    segunda(S, S1, 0);
}


/*
 * Pre: (PT alfa EN [inf,sup-1]. S.data[alfa] <= S.data[alfa+1])
 * Post: tercera(S,minimo,inf,sup) = (NUM alfa EN [inf,sup]. S.data[alfa] >= minimo)
 */
template <typename T>
int tercera (const Secuencia<T> S, const T minimo, const int inf, const int sup) {
    if (inf < sup) {
        int medio = (inf + sup) / 2;
        if (S.data[medio] >= minimo) {
            return sup - medio + 1 + tercera (S, minimo, 0, medio - 1);
        }
        else {
            return tercera (S, minimo, medio + 1, sup);
        }
    }
    else {
        if (S.data[inf] >= minimo) {
            return 1;
        }
        else {
            return 0;
        }
    }
}

/*
 * Pre: (PT alfa EN [0,S.n-2]. S.data[alfa] <= S.data[alfa+1])
 * Post: tercera(S,minimo) = (NUM alfa EN [0,S.n-1]. S.data[alfa] >= minimo)
 */
template <typename T>
int tercera (const Secuencia<T> S, const T minimo) {
    return tercera (S, minimo, 0, S.n - 1);
}
```