Escuela de Ingeniería y Arquitectura - Depto. de Informática e Ingeniería de Sistemas
Trabajo obligatorio de Programación 2 - Alumnos del grupo de tardes - Junio de 2016

La evaluación de la asignatura Programación 2 en la convocatoria de junio de 2016 consta de tres pruebas:

- Examen escrito del 8 de junio de 2015 con un peso del 70 %. Para aprobar la asignatura es necesaria una calificación mínima de 4 puntos en él.
- Examen práctico en laboratorio del 8 de junio de 2015 con un peso del 15 %.
- El presente trabajo obligatorio, con un peso del 15 %, lo entregarán exclusivamente los alumnos matriculados en el grupo de tarde al comienzo del examen escrito del 8 de junio de 2016 o en los días previos.

El tipo genérico **`Secuencia`**, con el que se trabajó en la práctica 5ª, se ha definido aquí del siguiente modo.

```
/*
 * Número máximo de elementos en una secuencia (un valor positivo)
 */
const int MAX = ... ;          // Definir su valor según necesidades (siempre MAX > 0)

/*
 * Un dato definido a partir del tipo genérico Secuencia representa una secuencia
 * de elementos de tipo T
 */
template <typename T>
struct Secuencia {
    // El valor de numDatos define el número K de elementos de la secuencia [d_1, d_2, ..., d_K]
    // con K >= 0 y K <= MAX
    int numDatos;
    // Los elementos de la secuencia [d_1, d_2, ..., d_K] se almacenan de forma contigua
    // en el vector datos:
    //     d_1 se almacena en datos[0]
    //     d_2 se almacena en datos[1]
    //      ...
    //     y, finalmente, d_K se almacena en datos[K-1]
    T datos[MAX];
};
```

Para trabajar con datos definidos a partir del tipo genérico **`Secuencia`** se propone diseñar las cinco funciones genéricas que se especifican a continuación. El diseño de la función **`retirarPrimero`**`(S)` ha de plantearse de forma que ni en su código ni, en su caso, en el código de sus funciones auxiliares, se haya programado ningún bucle. Por el contrario, en el código de las restantes funciones pedidas y, en su caso, en el código de sus funciones auxiliares, no puede programarse ninguna invocación recursiva.

```
/*
 * Pre:  cierto
 * Post: S.numDatos = 1 AND S.datos[0] = elemento
 */
template <typename T>
void unitaria (Secuencia<T>& S, const T elemento) {
     < escribir aquí el código de la función; ni su código ni, en su caso, el de
       sus funciones auxiliares pueden presentar invocaciones recursivas >
}

/*
 * Pre:  S.numDatos = K AND K >= 0 AND K < MAX AND
 *       (PT alfa EN [1,K]. S.datos[ alfa - 1] = Do[alfa])
 * Post: S.numDatos = K + 1 AND S.datos[K] = ultimo AND
 *       (PT alfa EN [1,K]. S.datos[ alfa - 1] = Do[alfa])
 */
template <typename T>
void insertarUltimo (Secuencia<T>& S, const T ultimo) {
     < escribir aquí el código de la función; ni su código ni, en su caso, el de
       sus funciones auxiliares pueden presentar invocaciones recursivas >
}

/*
 * Pre:  S.numDatos = K AND K >=1 AND K <= MAX AND
 *       (PT alfa EN [1,K]. S.datos[ alfa - 1] = Do[alfa])
 * Post: S.numDatos = K - 1 AND
 *       (PT alfa EN [2,K]. S.datos[ alfa - 2] = Do[alfa])
 */
template <typename T>
void retirarPrimero (Secuencia<T>& S) {
     < escribir aquí el código de la función; ni su código ni, en su caso, el de
       sus funciones auxiliares pueden presentar bucles >
}

/*
 * Pre:  S.numDatos = K AND K >= 0 AND K <= MAX AND
 *       (PT alfa EN [1,K]. S.datos[ alfa - 1] = Do[alfa])
 * Post: S.numDatos = K AND (K > 0 -> S.datos[0] = Do[K]) AND
 *       (PT alfa EN [1,K-1]. S.datos[ alfa ] = Do[alfa])
 */
template <typename T> void rotarDerecha (Secuencia<T>& S) {
     < escribir aquí el código de la función; ni su código ni, en su caso, el de
       sus funciones auxiliares pueden presentar invocaciones recursivas >
}

/*
 * Pre:  S.numDatos = K AND K >= 0 AND K <= MAX
 * Post: longitud(S) = K
 */
template <typename T>
int longitud (const Secuencia<T> S) {
     < escribir aquí el código de la función; ni su código ni, en su caso, el de
       sus funciones auxiliares pueden presentar invocaciones recursivas >
}
```

Cada alumno matriculado en el **grupo de tardes** debe presentar en papel, impreso o escrito a mano, un trabajo realizado individualmente, que conste de los apartados que se describen a continuación:

1. Diseño (especificación formal y código `C++`) de las cinco funciones genéricas anteriores. Se recuerda que la función **`retirarPrimero`**`(S)` y, en su caso, sus funciones auxiliares, deben presentar un diseño sin bucles. En el diseño de las restantes funciones se permite la programación de bucles, en el caso de que sean necesarios. [0 puntos]

2. Demostración formal de la corrección de esas mismas cinco funciones genéricas y, en su caso, de sus funciones auxiliares. Cada demostración formal constará de un conjunto de cálculos y pruebas que podrán escribirse intercaladas sobre el propio código o aparte. En cualquier caso, las pruebas se presentarán de forma que sean fácilmente legibles. [10 puntos]