Introducción al Modelado y la Simulación
TAREA N° 2
Batería de pruebas de uniformidad e independencia
Fecha de entrega: martes 25 de febrero antes de las 6 p.m. por e-mail

Considere los siguientes generadores de números aleatorios:

$$X_n = 48271X_{n-1} \mod (2^{31}-1) \qquad r_n = \frac{X_n}{2^{31}-1}$$

$$Y_n = 69621Y_{n-1} \mod (2^{31}-1) \qquad s_n = \frac{Y_n}{2^{31}-1}$$

$$Z_n = 16807Z_{n-1} \mod (2^{31}-1)$$

Se debe efectuar pruebas de uniformidad usando Kolmogorov-Smirnov (KS) y de independencia usando correlación serial (CS) con desplazamiento entre 1 y 5, con **$\alpha = 0.05$,** al primer generador (X_n). Estas pruebas se le harán a muestras de este generador. El tamaño de una muestra será uniforme entre 35 y 60. Para generar una variable aleatoria uniforme entera en $[m,n]$ se usa $\lfloor m+(n-m+1)u \rfloor$ donde u es un número aleatorio. Este tamaño se generará usando la secuencia Y_n. Es decir, usaremos `Trunc(35+(60-35+1)`s_n`)` o `Trunc(35+26`s_n`)`. La semilla de la muestra se generará usando Z_n.

Con más detalle. Al comenzar el programa se pide el número de pruebas y las semillas de las secuencias Y_n y Z_n, es decir, Y_0 y Z_0 (ver ejemplo en la pagina siguiente). Para X_n no hay que pedir semilla ya que esta se generará usando Z_n. Luego, para las primeras pruebas generamos el tamaño de la muestra usando `Trunc(35+26`s_1`)` y usamos Z_1 como la semilla X_0. Generamos los valores de la muestra (r_n) y se hacen las pruebas. Para las segundas pruebas generamos el tamaño de la muestra usando `Trunc(35+26`s_2`)` y usamos Z_2 como la semilla para la nueva muestra de los *X's y r's*. Se hacen las pruebas. Así sucesivamente.

Los 3 generadores deben ser implementados usando el método de **Schrage**, tal como se vio en clase y aparece en las notas, en donde se expresa $ax \mod m$ como $g(x) + mh(x)$ … (pagina IV-5).

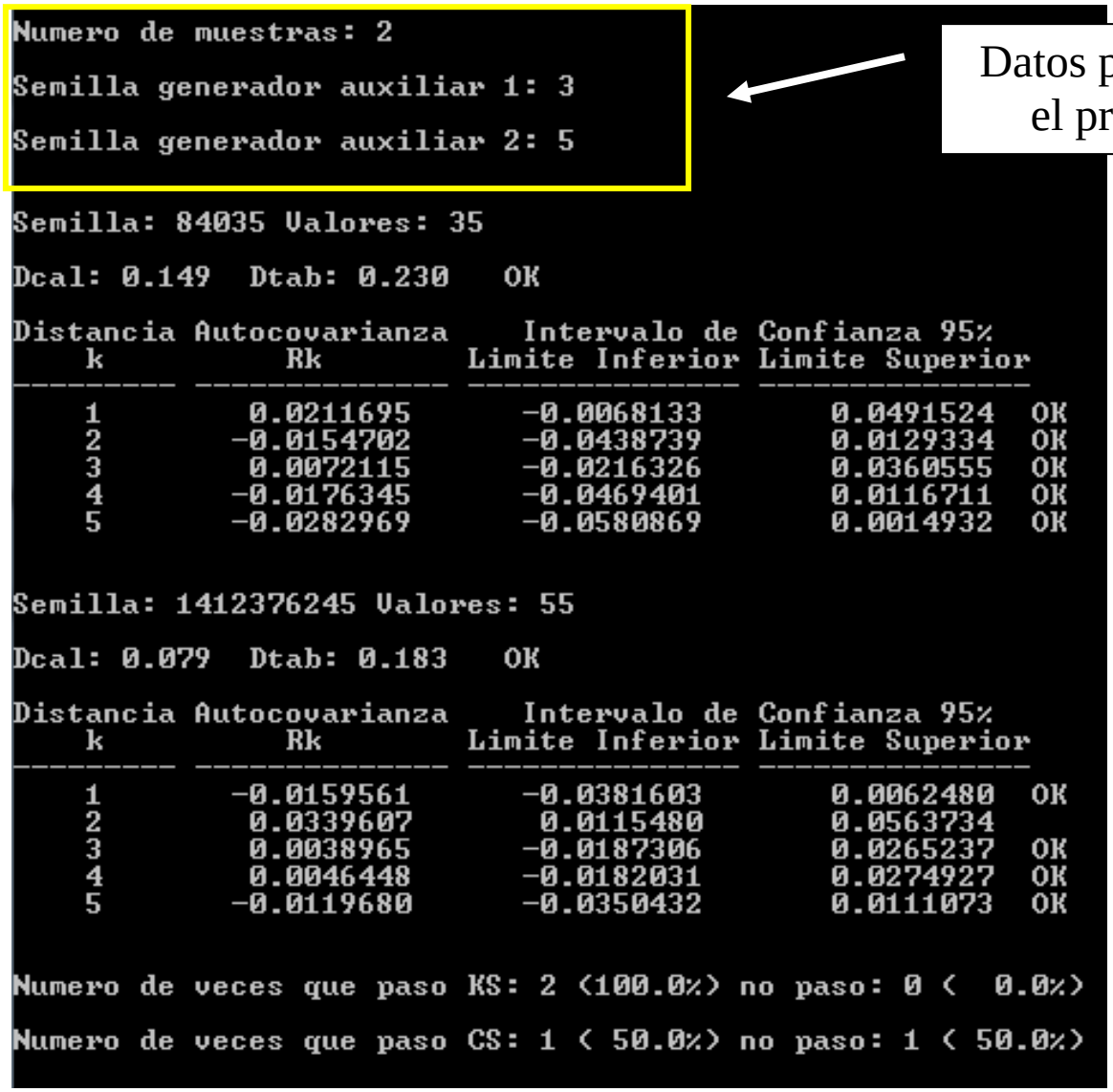
Dado que se debe ordenar la muestra para hacer la prueba K-S, pueden usar:

```
procedure sort(var v:array of real; n:integer);
{ Metodo más primitivo para ordenar un arreglo
  v: es dirección en memoria del vector a ser ordenado
  n: es el numero de elementos a ser considerados en el vector.
     Ojo, independientemente de como este definido el vector que se este pasando
     como parámetro, dentro de la rutina sort el primer elemento es v[0], ya que
     no se esta especificando el rango de los índices. Por ejemplo, si
     externamente se esta definiendo datos: array[10..50] of real y luego se
     llama sort(datos,20), el primer elemento dentro de sort es v[0] y no v[10],
     y solo se ordenaran los primeros 20 elementos de datos}

var i,j:integer;
    x:real;
begin
  for i:=0 to n-2 do
    for j:=i+1 to n-1 do
      if v[i]>v[j] then begin
        x:=v[j];
        v[j]:=v[i];
        v[i]:=x;
      end;
end;
```

Hay que mostrar los resultados de las pruebas e indicar cuantas fueron pasadas y cuantas no tal como se muestra en la figura siguiente (usted debería obtener exactamente los mismos valores mostrados para las entradas dadas-

las 3 primeras líneas). Dcal es el valor calculado de la prueba y Dtab es el valor teórico que por ser las muestras mayores a 35 se calcula fácilmente como $1.36/\sqrt{n}$. Para la prueba de correlación serial, si uno de los intervalos de confianza para alguno de los desplazamientos no incluye el cero, considere que la muestra no pasa la prueba.



Se debe enviar solo el código fuente (**.dpr),** a Herbert Hoeger: hhoeger@ula.ve. ***NOTA***: no enviar ejecutables (**.exe**) ya que los servidores de la ULA no dejan pasar estos correos. Recuerde que:

- **Cuando se reciba la tarea, se le devolverá un correo indicando que la misma fue recibida. Esta respuesta es manual y no automatizada (paciencia por favor). Mantenga este correo (no lo borre) ya que es el único medio de probar que fue enviada. Sin el no hay posibilidad de reclamar que usted la envió y no fue corregida.**
- **Los códigos deben cumplir con estilo de programación (principalmente identación)**
- **Estar debidamente identificado (autor del mismo) y documentado.**