Programación Concurrente y de Tiempo Real
Grado en Ingeniería Informática
Examen Final de Prácticas
5 de Septiembre de 2022

1. Notas de Procedimiento

1. Dispone de 100 minutos para completar el ejercicio.
2. Puede utilizar el material bibliográfico (libros) y copia de API que estime convenientes.
3. Entregue sus productos, utilizando la tarea de entrega disponible en el Campus Virtual, en un fichero (`.rar, .zip`) de nombre

   `Apellido1_Apellido_2_Nombre`

   que contendrá una subcarpeta por cada enunciado del examen, la cual contendrá a su vez el conjunto de ficheros que den solución a ese enunciado en particular. Cada fichero de código debe incluir un comentario con su nombre, apellidos y D.N.I.
4. Los productos de examen deben entregarse **única y exclusivamente** mediante la tarea de entrega habilitada para ello en el Campus Virtual, y dentro del plazo de tiempo establecido. Entregas efectuadas incumpliendo lo anterior no serán consideradas a efectos de corrección.

2. Criterios Generales de Corrección

Los códigos entregados deben satisfacer los siguientes criterios[1]:

- compilar correctamente sin errores ($-n$);
- ejecutarse sin lanzamiento de excepciones u otras anomalías graves de ejecución ($-n$);
- poseer una semántica que da soporte a la solución pedida en términos de entrada/salida, seguridad, vivacidad, rendimiento, etc. ($-n$);

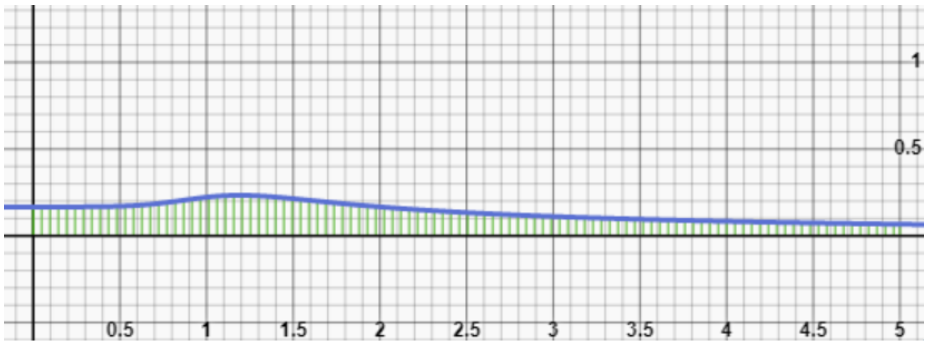
[1] Entre paréntesis, se indica la puntuación a descontar derivada del incumplimiento del criterio, para un enunciado valorado con n puntos.

- recoger íntegramente todas las especificaciones establecidas en el documento de examen e implementarlas tal y como se especifican ($-2{,}5$/especificación).
- estar contenidos en ficheros conformes en número, formato y nomenclatura de nombres con lo explicitado en este documento de examen ($-n$).

3. Enunciados de Examen

1. (5.5 puntos, Integración Paralela) Se desea disponer de un programa que calcule, aplicando el método de Monte-Carlo, la siguiente integral definida (el desarrollo de la función en el intervalo de integración $[0, 5]$ se ilustra en la Figura):

$$\int_0^5 \frac{x^5 + 1}{(3x^6 + 6)} dx$$



Desarrolle utilizando el lenguaje Java un programa que permita calcular la integral de forma paralela de acuerdo al siguiente conjunto de especificaciones:

- No utilice paquetes (`package`);
- Guarde el código en `defIntegration.java`.
- Tareas paralelas modeladas mediante herencia de la clase `Thread`. Todas las tareas se almacenarán en un array `tasks`. Cada tarea calcula la subsuperficie correspondiente a un subintervalo de integración distinto de forma independiente, aplicando en el mismo el método de Monte-Carlo. AYUDA: la función a integrar está acotada superiormente en todo el intervalo de integración por uno (ver Figura).
- Cada tarea recibe su subintervalo de trabajo mediante dos parámetros `linf` y `lsup` que se transfieren a través del constructor.
- Cada tarea acumula la subsuperficie que ha calculado en una variable compartida de tipo `double` llamada `areaParcial`, que al finalizar la ejecución contendrá la superficie (integral) buscada.
- Habrá exactamente cinco tareas paralelas.

- La variable `areaParcial` estará protegida por un protocolo de control de exclusión mutua que utilizará, obligatoriamente, `synchronized`.
- Cada tarea debe disponer de su propio generador de números aleatorios de clase `Random`, independiente de los demás; los generadores estarán inicializados con diferentes semillas obligatoriamente.
- Cada una de las cinco tareas paralelas utilizará un total de 3×10^6 puntos aleatorios. La tarea secuencial utilizará 15×10^6 puntos aleatorios.
- El programa principal gestionará las hebras mediante los métodos `start()-join()`.
- El programa calculará la integral de forma secuencial y paralela, y terminará imprimiendo los tiempos en nanosegundos de cálculo secuencial y paralelo, los valores de la integral calculada secuencial y paralelamente, y finalmente el *speedup*, con un *output* como el siguiente:

```
Tiempo secuencial: n nanosegundos
Tiempo paralelo: m nanosegundos
Integracion secuencial: p (p es el valor calculado)
Integración paralela: q (q es el valor calculado)
Speedup: s
```

2. (4.5 puntos, MPJ-Express, Token-Ring) Deseamos disponer de un programa escrito en Java utilizando la interfaz de paso de mensajes MPJ-Express de acuerdo a las siguientes especificaciones:

   - Habrá un total de diez procesos en tiempo de ejecución.
   - Los procesos utilizarán las operaciones de envío y recepción de mensajes `Send/Receive` para construir una topología de los procesos en anillo.
   - Habrá una ficha (token) que se irán enviando los procesos de uno a otro, comenzando por el proceso con `rank` igual a cero. La ficha tendrá inicialmente un valor igual a diez, y **cada proceso que la reciba** la incrementará en uno antes de enviarla al siguiente proceso del anillo. La existencia, envío e incremento del token deben ser explícitos, y su valor no debe depender en modo alguno del `rank` de los procesos. El token deberá dar dos vueltas completas a lo largo del anillo.
   - El output en tiempo de ejecución del programa deberá ser el siguiente:

```
Soy el proceso 0 y el token vale: 10
Soy el proceso 1 y el token vale: 11
Soy el proceso 2 y el token vale: 12
Soy el proceso 3 y el token vale: 13
Soy el proceso 4 y el token vale: 14
Soy el proceso 5 y el token vale: 15
Soy el proceso 6 y el token vale: 16
Soy el proceso 7 y el token vale: 17
Soy el proceso 8 y el token vale: 18
Soy el proceso 9 y el token vale: 19
Soy el proceso 0 y el token vale: 20
```

```
Soy el proceso 1 y el token vale: 21
Soy el proceso 2 y el token vale: 22
Soy el proceso 3 y el token vale: 23
Soy el proceso 4 y el token vale: 24
Soy el proceso 5 y el token vale: 25
Soy el proceso 6 y el token vale: 26
Soy el proceso 7 y el token vale: 27
Soy el proceso 8 y el token vale: 28
Soy el proceso 9 y el token vale: 29
```

- Incluya en su programa como comentarios las órdenes de línea de comando necesarias para compilar y ejecutar su programa. Esta última debe incluir el valor del flag `-np` para crear el anillo de procesos que propone. La ausencia de estas órdenes invalida por completo el ejercicio, con independencia de los criterios generales de corrección.
- Guarde el código en `tokenRing.java`.