Trabajo Práctico N° 1- Parte2

Procesos livianos (Threads)

Fecha de entrega:

a. 14/05/2025

Forma de entrega:

b. Se deberá generar un informe que contenga los siguientes puntos
   - **Carátula**: con los integrantes del grupo.
   - **Link a un repositorio de github:** Generar un archivo de Colab por cada lenguaje utilizado y almacenarlos en este repositorio. Estos archivos deben ser con la extensión ipynb. También se debe subir a github el código fuente.
   - **Conclusiones:** En esta sección se debe describir las dificultades que encontraron al realizar el trabajo práctico.

c. Entregar el informe por plataforma MIeL. Este debe ser en formato .pdf, con nombre TP1_P2_NumerodelGrupo.pdf.

Enunciado:

Para cada uno de los siguientes enunciados, elija un lenguaje entre **C++, Python o JAVA,** sin repetir el lenguaje, y resuelva lo pedido.

a. El entrenamiento de Naruto Uzumaki

Naruto Uzumaki dispone de poco tiempo para subir de nivel su técnica “Rasengan” antes de que llegue la siguiente amenaza a la aldea. Para acelerar el proceso de entrenamiento utilizará su Jutsu “Multi Clones de Sombra”, permitiendo que cada clon realice una parte del entrenamiento y al final Naruto incorpore el avance de cada clon como propio.

Sobre los clones se sabe lo siguiente:

- La cantidad de Chakra (energía) para entrenar es variable (5 a 10 unidades).
- Al consumir todo el Chakra el clon desaparece finalizando su sesión de entrenamiento.
- La sesión de entrenamiento de cada clon se compone de varios intentos (uno por cada unidad de Chakra).
- La duración de un intento es variable (100 a 200 milisegundos).
- Cada intento puede salir bien (se sube un punto de nivel) o salir mal (no se sube de nivel). Hay una probabilidad del 0.5 de subir de nivel.

Se pide:

- Implementar un programa que simule el entrenamiento de Naruto utilizando programación concurrente a través de N hilos, permitiendo tomar el tiempo de ejecución.
- No utilizar ningún mecanismo de comunicación ni sincronización entre los hilos.

- Ejecutar el programa para distintas cantidades de clones (5, 10, 20, etc). Informando el tiempo y nivel alcanzado para cada ejecución.
- Responda: ¿Cuál sería la cantidad óptima de clones? Justifique su respuesta.
- Realizar un gráfico de tiempo en base a la cantidad de clones.

b. <u>El Gran Director</u>

Posiciones Finales Apertura 1992

#	Equipo	Pts	PJ	PG	PE	PP	GF	GC	DIF
1	Boca Juniors	27	19	10	7	2	24	11	13
2	River Plate	23	19	10	5	4	28	13	15
3	San Lorenzo	23	19	9	5	5	28	19	9
4	Ferro	22	19	6	10	3	16	9	7
5	Huracan	22	19	9	4	6	26	22	4
6	Velez	21	19	8	5	6	23	15	8
7	Estudiantes (LP)	20	19	7	6	6	21	14	7
8	Belgrano	20	19	7	6	6	22	22	0
9	Lanus	20	19	8	4	7	22	22	0
10	Talleres (C)	20	19	6	8	5	18	20	-2
11	Dep Español	19	19	7	5	7	19	18	1
12	San Martin (T)	18	19	6	8	5	18	14	4
13	Dep Mandiyu (C)	18	19	5	8	6	21	24	-3
14	Rosario Central	18	19	7	4	8	19	29	-10
15	Independiente	17	19	5	7	7	15	22	-7
16	Racing Club	15	19	4	7	8	14	20	-6
17	Gimnasia (LP)	15	19	4	7	8	19	27	-8
18	Platense	14	19	3	8	8	16	21	-5
19	Argentinos	14	19	3	8	8	17	25	-8
20	Newells	10	19	3	4	12	12	31	-19

A River se le descontaron 2 puntos por incidentes en la última fecha.
A San Martín (T) se le descontaron 2 puntos por incidentes.
Se computaban 2 puntos por partido ganado.

Un pequeño equipo de desarrollo de videojuegos, está haciendo pruebas de concurrencia para su próximo videojuego; *El Gran Director*. Actualmente se encuentran trabajando en el módulo de simulación de partidos tratando de reducir los tiempos de simulación de cada fecha. El equipo lo contrata a usted, experto en programación concurrente para evaluar cuanto mejoraría (si es que mejora) la simulación al utilizar concurrencia en contraste con una simulación donde la ejecución se realice de manera secuencial.

El equipo de desarrollo proporciona la siguiente información:

- Actualmente están trabajando con la tabla del torneo Apertura 1992 para realizar las pruebas, con lo cual los equipos deberían ser los mismos que figuran en dicha tabla.
- La generación del fixture es por medio del sistema “todos contra todos” o Round Robin. Dado que es un algoritmo conocido, el equipo considera que usted puede programarlo fácilmente así que no le proporcionan el código del mismo, solo el siguiente enlace: https://es.wikipedia.org/wiki/Sistema_de_todos_contra_todos#
- La información estadística a generar para cada equipo es la misma que figura en la tabla (Pts, PJ, PG, PE, PP, GF, GC, DIF).
- El criterio para determinar ganador, perdedor o empate queda a cargo del grupo. El único dato a tener en cuenta es que la simulación de cada partido dura entre 0.1 y 0.15 segundos.
- Si bien la simulación se realiza por fecha, le pide a usted que realice la simulación para todas las fechas (solo de ida).

<u>Se pide:</u>

- Implementar una función que genere el fixture.
- Implementar un programa que en base al fixture generado, simule los partidos de todas las fechas (solo de ida), generando la información de la tabla.
- Para cada fecha; crear un hilo por partido, simular cada partido y al finalizar todos los hilos actualizar la tabla para esa fecha. Así sucesivamente con todas las fechas.
  (Si este punto se puede implementar con pool de hilos (Thread Pool) sería ideal, para poder reutilizar los hilos)

- Al finalizar la ejecución mostrar la tabla de posiciones final por pantalla y el tiempo insumido total.
- También implemente otro programa (o puede realizarlo en el mismo) que simule la ejecución secuencial con la misma salida por pantalla.
- No utilizar comunicación ni sincronización entre los hilos.

c. Análisis de preferencia

Una plataforma de Streaming está pensando en incorporar un algoritmo para poder detectar la preferencia de sus usuarios en cuanto a géneros de películas y series para poder recomendar contenido de manera personalizada. Los dueños de la plataforma contactan con usted para que presente una propuesta que cumpla con el objetivo en cuestión utilizando programación concurrente.

La plataforma dispone de la siguiente información almacenada en un archivo:

- visualizaciones.csv: actualmente se mantiene todo el contenido visto por todos los usuarios en un solo archivo (no ordenado), donde se almacena la siguiente información por línea:

  user_id,user_name,title,type,genre

  Por ejemplo:
  user_id,user_name,title,type,genre
  1,p_kiki,Hereditary,Película,Terror
  2,daro123,La Casa de Papel,Serie,Drama
  1,p_kiki,The Office,Serie,Comedia

Se pide:

- Implementar un programa que utilice programación concurrente mediante hilos, sin necesidad de comunicarse ni sincronizarse entre sí, donde cada hilo analizará las preferencias de un usuario en particular generando como resultado la información necesaria para que el proceso principal pueda generar el archivo preferencias.json con la siguiente información por usuario:

```
{
 "user_id": 1,
 "user_name": "p_kiki",
 "chosen_genre": "Terror",
 "chosen_type": "Película",
 "total": 2,
 "different_genres": 2
}
```

En caso de que exista algún tipo de “empate” y no se pueda decidir claramente por un género o tipo de contenido queda a criterio del grupo cual tomar como preferencia del usuario en cuestión.