**แบบฝึกหัด บทที่ 4**

1. เรื่อง User-Level Thread และ Kernel-Level Thread

* จงอธิบายความแตกต่างระหว่าง User-Level Thread และ Kernel-Level Thread

ตอบ 1.    User-level threads

           ชนิดการทำงานของเธรดใน User-level threads จะเป็นแบบ Many-to-one หมายถึง ระบบปฏิบัติการจะวางเธรดทั้งหมดของ Multithread process ไว้ใน Execution context เพียงตัวเดียว ทำงานอยู่ภายใต้พื้นที่ของผู้ใช้งาน ไม่เกี่ยวข้องกับพื้นที่ของเคอร์เนล จึงทำให้ไม่สามารถเข้าใช้งานเคอร์เนลได้โดยตรง

2.    Kernel-level threads

           ชนิดการทำงานของเธรดใน Kernel-level threads จะเป็นแบบ One-to-one หมายถึง ระบบปฏิบัติการจะวางเธรดแต่ละตัวไว้ใน Execution context ของตัวเอง ดังนั้น ในการวางเธรดแต่ละตัวจึงต้องให้ระบบปฏิบัติการช่วยในเรื่องการ Mapping ตัว User-level threads แต่ละตัวเข้ากับ Kernel-level threads

* สถานการณ์ใดที่เหมาะสมกับการใช้ User-Level Thread และ Kernel-Level Thread

ตอบ สถานการณ์แบบ many to many และ Two level การนำเอา one to one ของ Kernel level รวมกับ many to one ของ User level thread มีการจัดการแบบแบ่ง thread ที่เกิดขึ้นกระจายตัวให้กับ kernel โดย kernel หนึ่งตัวสามารถรอบรับ thread ได้มากกว่า 1 หากเป็น Two level จะมีการเพิ่มในการใช้ one to one สำหรับงานที่มีความสำคัญสูงและยังต้องการให้ kernel ที่เหลือทำงานจาก thread อื่น ๆไปด้วย

* สถานการณ์ใดที่เหมาะสมกับการใช้ Kernel-Level Thread

ตอบ สถานการณ์แบบ one to one Kernel ที่แต่ละ Kernel จะจับคู่กับ Thread แบบ 1 ต่อ 1

เหมาะกับการทำงานที่ต้องการความถูกต้องมีความสำคัญสูงแต่มีประสิทธิภาพในการทำงานช้า

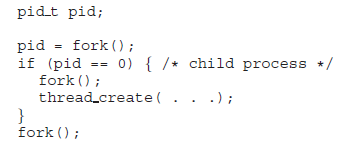
1. สถานการณ์ใดที่ Multithreaded Solution ที่ใช้ Multiple Kernel Thread ทำให้ระบบมีประสิทธิภาพดีกว่าการใช้ Single-Threaded Solution บนระบบที่มีโพรเซสเซอร์เดียว จงยกตัวอย่างพร้อมคำอธิบาย

ตอบ เมื่อ kernel thread ประสบปัญหาความผิดพลาด สามารถเปลี่ยนเป็น kernel thread อื่นได้ ในขณะเดียวกันกระบวนการแบบ Single-Threaded จะไม่สามารถทำงานที่ได้เมื่อเกิดข้อผิดพลาด ดังนั้นในสถานการณ์ที่โปรแกรมอาจประสบกับความผิดพลาดบ่อย วิธีแก้ปัญหาแบบ Multithreaded Solution จะทำงานได้ดีกว่าแม้ในระบบ Single-Threaded

1. สิ่งใดต่อไปนี้ Register Values , Heap Memory , Global Variables , Stack Memory ที่ถูกแชร์ระหว่าง Multithreaded Process

ตอบ Global Variables และ Heap Memory ที่ถูกแชร์ระหว่าง Multithreaded Process

1. จากส่วนของโค้ดต่อไปนี้



* มีโปรเซสที่สร้างขึ้นกี่โปรเซส

ตอบ มีโปรเซสที่สร้างขึ้น 6 โปรเซส

* มี Thread ที่สร้างขึ้นกี่ Thread

ตอบ มี Thread ที่สร้างขึ้น 8 Thread

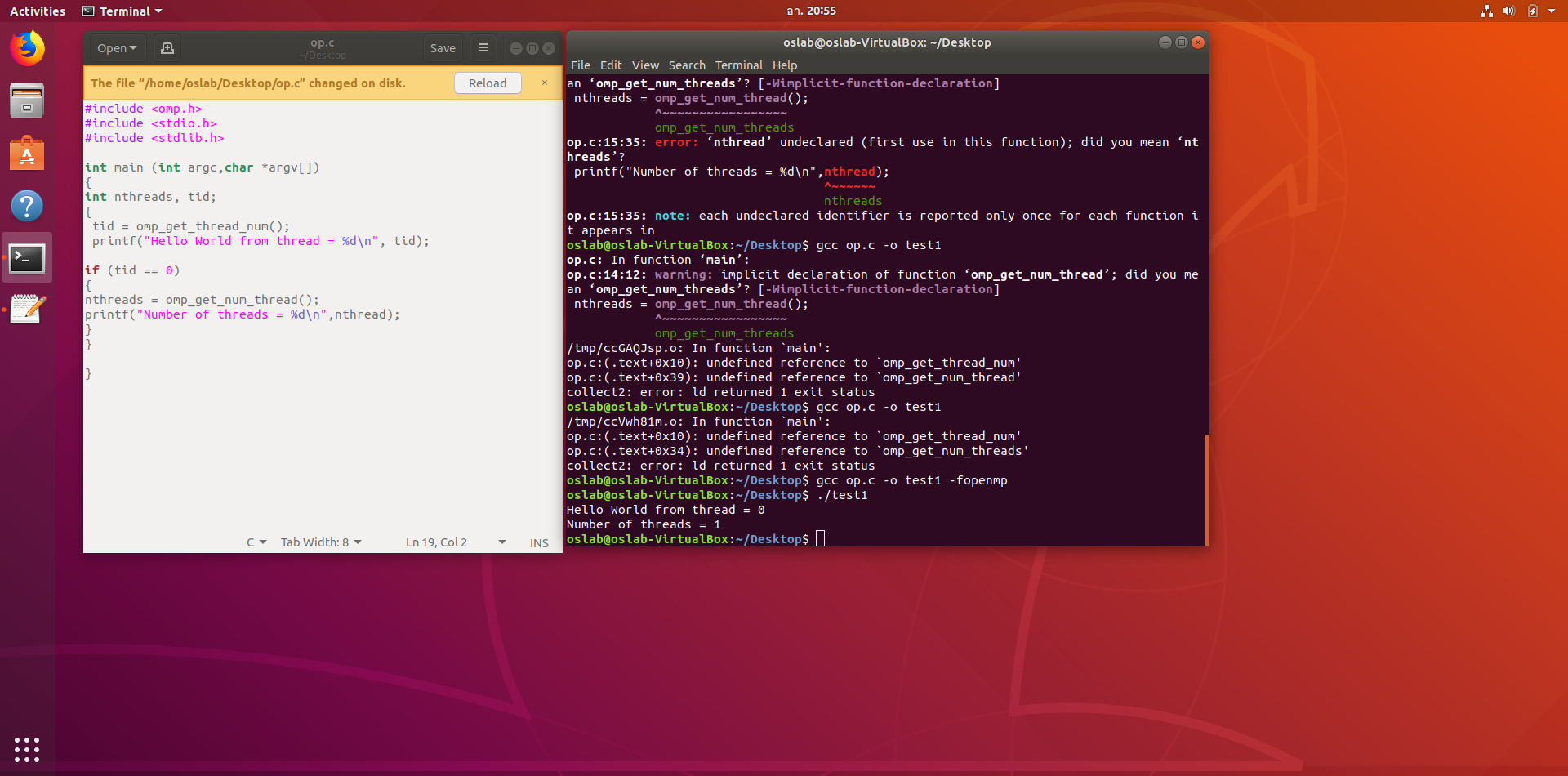
1. Thread Pool คืออะไร จงอธิบาย

ตอบ Thread Pool เป็น Class ที่ .NET Framework เตรียมเอาไว้ให้สำหรับการทำงานแบบ Multi-thread ที่ง่ายๆ ไม่ซับซ้อน รวมถึงการจัดการกับการทำงานของ Method ที่ Run แบบ Asynchronous ด้วย Thread Pool นั้น Run แบบ Background Thread ดังนั้น Application สามารถหยุดการทำงานได้ทันทีถ้า Foreground Thread อื่น ๆ สิ้นสุดการทำงานลง

Thread Pool นั้นเหมาะสำหรับที่จะใช้จัดการ Thread ที่ทำงานเล็ก ๆ ไม่ยุ่งยากซับซ้อนและใช้เวลาในการประมวลผลไม่นานนัก รวมทั้งไม่คำนึงถึงลำดับความสำคัญของ Thread อีกด้วย

1. จงหาตัวอย่างของโปรแกรมที่เขียนด้วย Java Thread และ OpenMP แสดงโค้ดพร้อมผลลัพธ์ของการรันที่ได้ และอธิบายพฤติกรรมของโปรแกรม

ตอบ OpenMP Code Sample code



( ภาพประกอบ Sample code ของ OpenMP และผลการทดลอง)

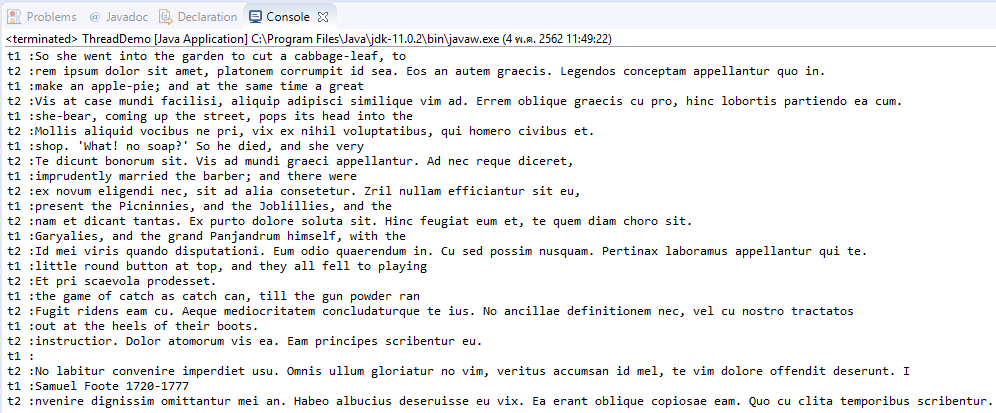
พฤติกรรมของโปรแกรม

โปรแกรมทำการรับ Thread ID ของ Thread มาเก็บที่ tid และหลังจากนั้นใช้ Thread ID ที่เก็บไว้ใน tid นั้นปริ้นข้อความออกมา ซึ่งจำนวนข้อความนั้นมีค่าเท่ากับตามจำนวนของ Thread ที่มีทั้งหมดและที่ Master Thread นั้นจะทำการนับค่าจำนวน Thread และเก็บไว้ใน nthreads และนำค่าในนั้นปริ้นแสดงผลออกมา

Java Thread Sample code



( ภาพประกอบ Sample code ของ JAVA)

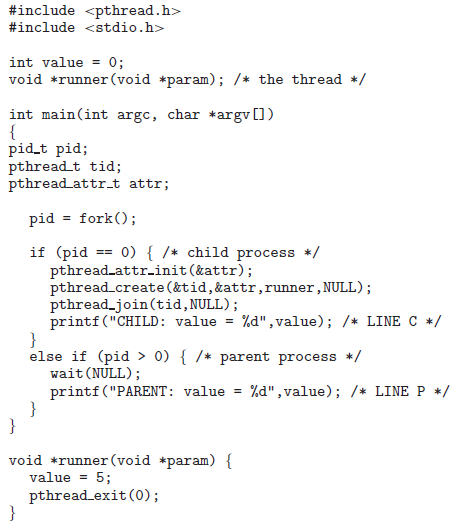


( ภาพประกอบจากผลการทดลอง )

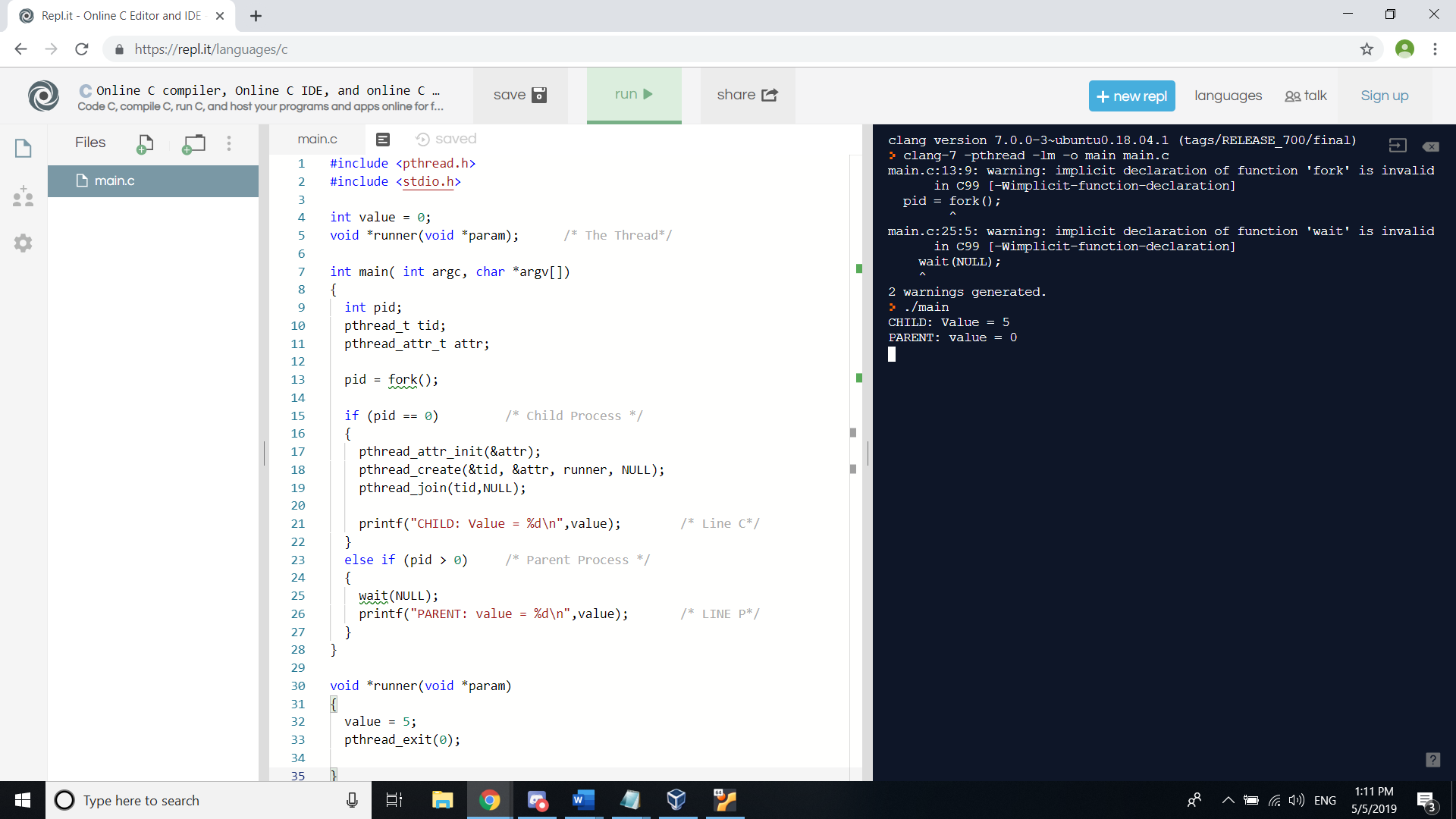
พฤติกรรมของโปรแกรม

โปรแกรมทำการสร้าง Thread ขึ้นมา 2 Thread มีการเรียกใช้ Thread ทั้ง 2 Thread โดยตรงจาก Class Reader1 และ 2 ตามลำดับ Thread ทั้ง 2 มีการทำงานเหมือนกันคืออ่านค่าจากไฟล์ text แต่เป็น text ที่ต่างกันทั้งคู่ทำงานไปในเวลาพร้อม ๆกันและทำให้ผลลัพธ์ที่จะแสดงผลค่าที่อ่านได้ออกมาเป็นบรรทัดจะเห็นได้ว่ามีการสลับไปสลับมาอยู่เกือบตลอดเนื่องจาก Thread ไหนทำเสร็จก่อนก็จะปริ้นแสดงออกมาก่อน

1. ทดลองโค้ดภาษาซีต่อไปนี้ ผลลัพธ์ที่ได้จาก Line C และ Line P เป็นอย่างไร



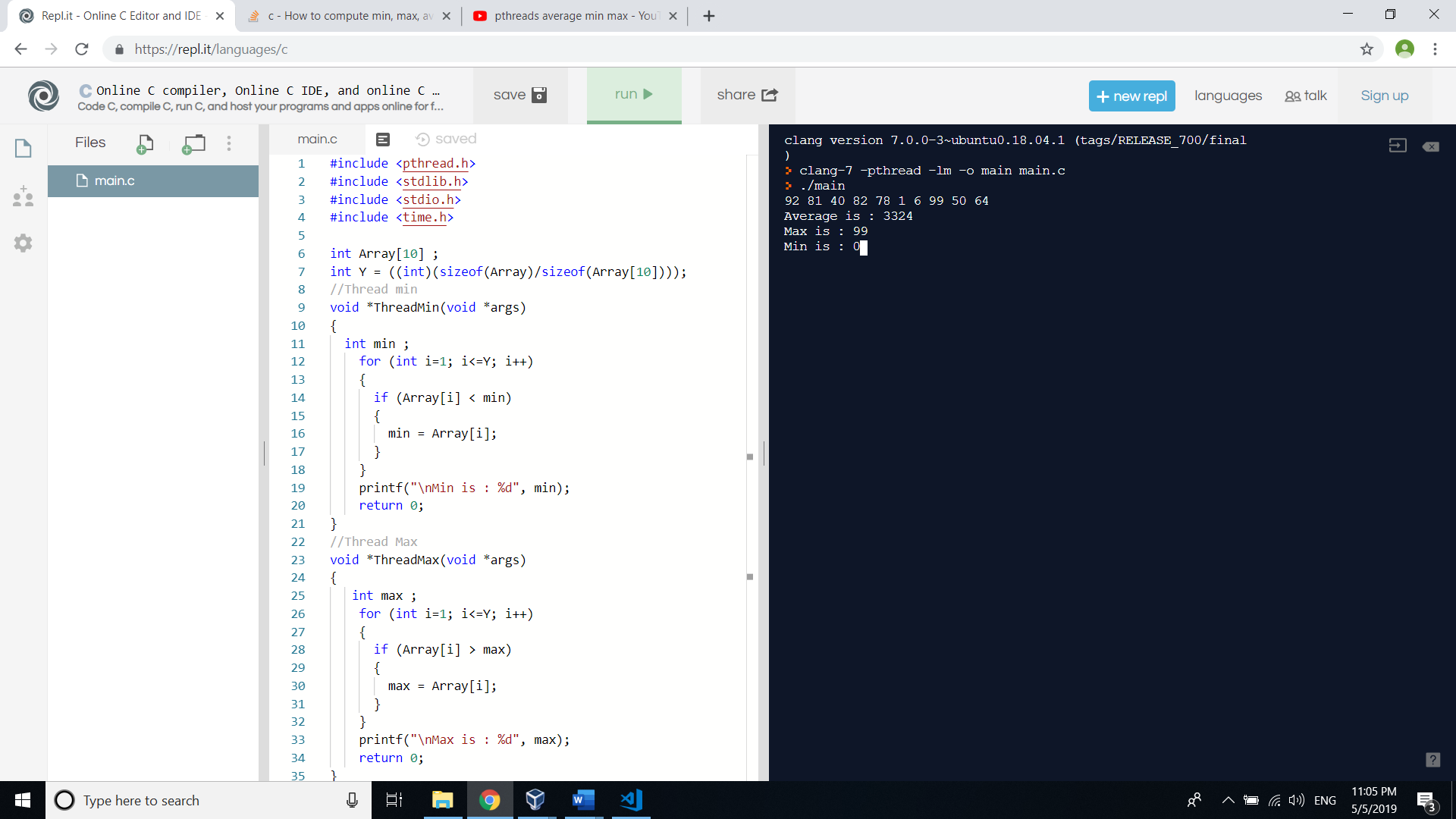
ตอบ Child process ถูก fork โดย parent process หลังจาก fork แล้ว parent process จะรอ child process รันจนเสร็จสมบูรณ์มี Thread ใหม่ที่ถูกสร้างขึ้นสำหรับ child process และเรียกใช้ฟังก์ชัน runner() ซึ่งตั้งค่าของ value เป็น5หลังจากเสร็จสิ้นกระบวนการของ child process ค่าของ value ที่มีอยู่ใน parent process จะเป็น 0



1. จงเขียนโปรแกรมแบบ Multithread โดยทีการทำงานดังนี้

* Main Thread สร้างเลขจำนวนเต็มแบบสุ่ม 10 ค่า แล้วแสดงออกทางจอภาพ
* Main Thread เตรียมตัวแปร min , max และ avg สำหรับเก็บค่าต่ำสุด , สูงสุด , และค่าเฉลี่ยตามลำดับ
* สร้าง Thread ย่อย 3 Thread มีหน้าที่ดังนี้ จากเลขจำนวนเต็มที่สุ่มมาจาก Main Thread

1. Thread#1: ค้นหาค่าต่ำสุด เก็บลงใน min
2. Thread#2: ค้นหาค่าสูงสุด เก็บลงใน max
3. Thread#3: ค้นหาค่าเฉลี่ย เก็บลงใน avg

* Main Thread แสดงผลลัพธ์ของ min , max , avg ออกทางจอภาพ

