11 โปรแกรม version control มีประโยชน์อย่างไร

ระบบที่จัดเก็บการเปลี่ยนแปลงที่เกิดขึ้นกับไฟล์หนึ่งหรือหลายไฟล์ และยังช่วยให้ย้อนไฟล์บางไฟล์หรือแม้กระทั่งทั้งโปรเจคกลับไปเป็นเวอร์ชั่นเก่าได้

12.ข้อได้เปรียบของ distributed version control เมื่อเทียบกับ centralized version control คืออะไร

-distributed version control นั้นเมื่อระบบหรือเซิฟเวอร์ล่ม ก็จะสามารถที่จะทำงานได้ต่อ และยังก๊อปปี้ที่เซิร์ฟเวอร์เพื่อกูข้อมูลกลับคืนก็ได้ แต่ Centralized version control นั้นเมื่อเซิฟเวอร์ล่มไม่สามารถทำงานได้ต่อและถ้าฮาร์ดดิสเสียก็ไม่มีการสำรองข้อมูลอีกด้วย

13.ข้อได้เปรียบของ Centralized control เมื่อเทียบกับ distributed version control คืออะไร

-Centralized control จะได้เปรียบตรงที่ มีเซิฟเวอร์เป็นตัวกลางเก็บไฟล์ทั้งหมดไว้ที่เซิฟเวอร์ตัวเดียวและ User คนอื่นสามารถเรียกใช้ไฟล์ได้จาก Server ตัวนั้นได้อย่างง่าย

14.บอกแนวทางในการแก้ไข comflict ที่เกิดขึ้นเมื่อมีการ merge โปรแกรมของผู้พัฒนาหลายๆคนเข้าด้วยกัน  
 1.ปรึกษาคนในทีมว่ากำลังแก้ไข class อะไรกันอยู่

2.Mob programming ทุกคนจะได้มานั่งทำงานด้วยกัน ใช้เครื่องทำงานเดียวกัน

15..บอกแนวทางในการลดcomflict ที่เกิดขึ้นเมื่อมีการ merge โปรแกรมของผู้พัฒนาหลายๆคนเข้าด้วยกัน

1.ทำการ Merge บ่อยๆ 2-3ครั้ง/สัปดาห์

2.การพูดคุยื่อสาร

16.Git คืออะไร แตกต่างจาก Github อย่างไร

-git จะทำหน้าที่ในการจัดการเกี่ยวกับเวอร์ชันของซอร์สโค้ดของเรา ไม่ว่าจะเป็นเรื่องการตรวจสอบความแตกต่างระหว่างเวอร์ชัน (diff) การย้อนกลับมาเวอร์ชันก่อนหน้าเมื่อมีปัญหา (revert) หรือตรวจสอบเกี่ยวกับรายละเอียดของแต่ละเวอร์ชัน

-แต่ Github นั้นคือการฝากไฟล์ไว้บนเซิฟเวอร์

17.จุดประสงค์หลักในการ branch คืออะไร  
 - เพื่อแยกการพัฒนานั้น จะต้องทำการ merge การเปลี่ยนแปลงเข้ามายัง master branch หลังจากที่พัฒนาเสร็จแล้วเสมอ

18.Fast forward merge คืออะไรและทำไมการ push ไปที่ remote repo จึงควรจำต้อง merge แบบนี้

- ทำการ Merge Branch test ไปยัง master จะเป็นการ Merge แบบ Fast-forward

-push ความเปลี่ยนแปลงไปยัง remote repo

19.git pull ก็คือรวมโค๊ดจาก remote มายัง local โดยที่เราไม่สามารถรู้ได้เลยว่าจะรวมโค๊ดอะไรบ้าง รู้แค่หลังจาก pull เสร็จแล้วนั่นเอง ซึ่งจริงๆแล้ว git pull มันก็คือการทำ git fetch และต่อด้วย git merge อัตโนมัตินั่นเอง

20.workflow มันคือแนวทางที่จะทำให้การทำงานมันราบรื่นนั่นแหละ แต่มันมีแนวทางแนวทางหนึ่งที่เป็นที่นิยม เขาตั้งชื่อมันว่า gitflow

1. เมื่อจะพัฒนาฟีเจอร์ใหม่ ให้แตก branch Feature มาจาก Develop ให้ทีมพัฒนาโค้ดกันใน branch นั้น เมื่อพัฒนาเสร็จให้ merge โค้ดเข้า Develop แล้วลบ branch Feature ทิ้ง

2. เมื่อจะเอาโค้ดขึ้นโปรดักชั่น ให้แตก branch Release ออกมาจาก Develop ให้ Tester ตรวจสอบว่าโปรแกรมทำงานถูกต้องหรือเปล่า มีบักหรือเปล่า หากมีบักก็แก้ใน Branch Release เลย จนเมื่อโปรแกรมถูกต้องสมบูรณ์จึง merge เข้า Master เพื่อเอาขึ้นโปรดักชั่นต่อไป อย่าลืม Tag เวอร์ชั่น และ merge เข้า Develop ด้วยจากนั้นจึงลบ branch Release ทิ้ง

3. ทีนี้หลังจากที่เอาโค้ดขึ้นโปรดักษ์ชั่นแล้ว มันอาจจะพบบักที่ไม่คาดฝัน อาจจะเพราะ environment เครื่องหรืออะไรก็แล้วแต่ ซึ่งจำเป็นต้องรีบแก้ ให้แตก branch Hotfix ออกมาจาก Master แล้วแก้ไขบักซะ หลังจากแก้ไขเสร็จแล้ว ให้ merge โค้ดเข้าไปยัง Master แล้วลบ branch Hotfix ทิ้ง

จากที่กล่าวมา 3 ข้อด้านบน จะเห็นว่ามีการ สร้าง branch , merge branch, ลบ branch วนๆซ้ำไปซ้ำมา เวลาคุยกันในทีม มันก็กลายเป็นคำภาษาเทคนิคไป คนที่เขาเสนอแนวทาง gitflow ก็เลยสร้างชุดคำสั่งเสริมเข้าไปใน git เพิ่มเติม เพื่อให้ฟิลลิ่งมันระรื่น feature start, feature finish, release start, release finish, hotfix start, hotfix finish