**11. โปรแกรม version control มีประโยชน์อย่างไร**

* จัดเก็บการเปลี่ยนแปลงที่เกิดขึ้นกับไฟล์หนึ่งหรือหลายไฟล์เพื่อที่สามารถเรียกเวอร์ชั่นใดเวอร์ชั่นหนึ่งกลับมาดูเมื่อไรก็ได้

**12. ข้อได้เปรียบของ distributed version control เมื่อเทียบกับ centralized version control คืออะไร**

* มีเซิร์ฟเวอร์กลางที่เก็บไฟล์ทั้งหมดไว้ในที่เดียวและผู้ใช้หลาย ๆ คนสามารถต่อเข้ามาเพื่อดึงไฟล์จากศูนย์กลางนี้ไปแก้ไขได้ ทุกคนสามารถรู้ได้ว่าคนอื่นในโปรเจคกำลังทำอะไร ผู้ควบคุมระบบสามารถควบคุมได้อย่างละเอียดว่าใครสามารถแก้ไขอะไรได้บ้าง การจัดการแบบรวมศูนย์ในที่เดียวทำได้ง่ายกว่าการจัดการฐานข้อมูลใน client แต่ละเครื่องเยอะ

**13. ข้อได้เปรียบของ centralized version control เมื่อเทียบกับ distributed version control คืออะไร**

* แต่ละคนไม่เพียงได้ก๊อปปี้ล่าสุดของไฟล์เท่านั้น แต่ได้ทั้งก๊อปปี้ของ repository เลย หมายความว่าถึงแม้ว่าเซิร์ฟเวอร์จะเสีย client ก็ยังสามารถทำงานร่วมกันได้ต่อไป และ repository เหล่านี้ของ client ยังสามารถถูกก๊อปปี้กลับไปที่เซิร์ฟเวอร์เพื่อกูข้อมูลกลับคืนก็ได้ การ checkout แต่ละครั้งคือการทำสำรองข้อมูลทั้งหมด นอกจากนั้นระบบเหล่านี้ยังทำงานกับหลาย ๆ repository ได้อย่างดี ทำให้คุณสามารถทำงานกับคนหลายกลุ่มซึ่งทำงานในรูปแบบต่างกันในโปรเจคเดียวกันได้อย่างง่ายดาย เนื่องจากระบบเหล่านี้สนับสนุนการทำงานได้หลากหลายรูปแบบ

**14. บอกแนวทางในการแก้ไข conflict ที่เกิดขึ้นเมื่อมีการ merge โปรแกรมของผู้พัฒนาหลายๆคนเข้าด้วยกัน**

* ทำการ edit ส่วนที่เกิด conflict จะถูกคั่นด้วยบรรทัดที่ขึ้นต้นด้วยเครื่องหมาย <<<<<<< , ======= , >>>>>>> ในตัวอย่างนี้ จัดการลบ 3 บรรทดนั้นทิ้งไป แล้ว commit ไปใหม่

**15. บอกแนวทางในการลด conflict ที่เกิดขึ้นเมื่อมีการ merge โปรแกรมของผู้พัฒนาหลายๆคนเข้าด้วยกัน**

* 1.ทำการ Merge บ่อย ๆ ทุกครั้งเมื่อคุณทำการเปลี่ยนแปลง หรือ commit source code นั่นเอง  
  จะช่วยลดข้อขัดแย้งต่าง ๆ ลงไปอย่างมาก
* 2.พูดคุยกันในทีมก่อน บางครั้งต้องแก้ไข class เดียวกันอยู่ตลอดเวลา ควรนัดกันมาทำงานร่วมกันเพื่อลด conflict
* 3. การออกแบบระบบงาน ในหนึ่ง class ใน หนึ่ง method นั้นควรจะมีหน้าที่การทำงานเพียงอย่างเดียวเท่านั้นหรือในแต่ละ class แต่ละ method ควรมีเหตุผลเดียวในการเปลี่ยนแปลงเท่านั้น ผลที่ได้ก็คือ นักพัฒนาจะไม่ ทำงาน หรือ เปลี่ยนแปลง source code ที่เดียวกันอย่างแน่นอน

**16. Git คืออะไร แตกต่างจาก Github อย่างไร**

* Git คือ Version Control ตัวหนึ่ง ซึ่งเป็นระบบที่มีหน้าที่ในการจัดเก็บการเปลี่ยนแปลงของไฟล์ในโปรเจ็คเรา มีการ backup code ให้เรา สามารถที่จะเรียกดูหรือย้อนกลับไปดูเวอร์ชั่นต่างๆของโปรเจ็คที่ใด เวลาใดก็ได้ หรือแม้แต่ดูว่าไฟล์นั้นๆใครเป็นคนเพิ่มหรือแก้ไข หรือว่าจะดูว่าไฟล์นั้นๆถูกเขียนโดยใครบ้างก็สามารถทำได้ แต่ Github เป็นเว็บเซิฟเวอร์ที่ให้บริการในการฝากไฟล์ Git (ทั่วโลกมักนิยมใช้ในการเก็บโปรเจ็ค Open Source ต่างๆ ที่ดังๆ ไม่ว่าจะเป็น Bootstrap, Rails, Node.js, Angular เป็นต้น)

**17. จุดประสงค์หลักในการ branch คืออะไร**

* เป็น feature ที่ช่วยให้นักพัฒนาสามารถที่จะทำงานได้สะดวกขึ้น ยกตัวอย่างเช่น เรามีโค๊ดที่ดีอยู่แล้ว แต่อยากจะทดลองอะไรนิดๆหน่อย หรือแก้ไขอะไรก็ตาม ไม่ให้กระทบกับตัวงานหลัก ก็เพียงแค่สร้าง branch ใหม่ขึ้นมา เมื่อแก้ไขหรือทำอะไรเสร็จแล้ว ก็ค่อยเซฟกลับมาที่ master เหมือนเดิม

**18. Fast Forward Merge คืออะไรและทำไมการ push ไปที่ remote repo จึงควรต้อง merge แบบนี้**

* Fast Forward Merge จะเกิดขึ้นเมื่อเรา เป็นผู้ branch เพียงคนเดียว เมื่อเราทำการ merge ตัว git จะชี้ต้นแบบล่าสุดมาที่ branch ของเราแทน

**19. หน้าที่หลักของคำสั่ง git pull คืออะไร**

* git pull ก็คือรวมโค๊ดจาก remote มายัง local โดยที่เราไม่สามารถรู้ได้เลยว่าจะรวมโค๊ดอะไรบ้าง รู้แค่หลังจาก pull เสร็จแล้วนั่นเอง ซึ่งจริงๆแล้ว git pull มันก็คือการทำ git fetch และต่อด้วยgit merge อัตโนมัตินั่นเอง

**20. แผนภาพด้านล่างต้องการสื่อความหมายอะไร**

* ใน v0.1 มีการ branch ออกมาเพื่อพัฒนาและเพิ่ม Feature ต่างๆเข้ามากมาย เมื่อเพิ่มแล้วก็ได้ทำการ merge เข้ากับ Master เพื่อเป็นการอัพเดต Feature ต่างๆที่ถูกพัฒนาจาก v0.1 มาไว้ที่ v1.0