**Project 13-1: Greatest Common Divisor**

Create a program that finds the greatest common divisor of two numbers.

**Sample Run**

**Greatest Common Divisor**

**Number 1: 15**

**Number 2: 5**

**Greatest common divisor: 5**

**Continue? (y/n): y**

**Number 1: 15**

**Number 2: 6**

**Greatest common divisor: 3**

**Continue? (y/n): y**

**Number 1: 15**

**Number 2: 7**

**Greatest common divisor: 1**

**Continue? (y/n): n**

**Bye!**

**Specifications**

 Use the following recursive algorithm to calculate the greatest common divisor

(GCD):

**divide x by y and get the remainder**

**if the remainder equals 0, GCD is y (end function)**

**otherwise, calculate GCD again by dividing y by remainder**

 If number 1 is less than number 2, the program should display a message that

indicates that number 1 must be greater than number 2 and give the user another

chance to enter the numbers.

 The user will validate data.

**Project 13-2: Tree Pattern**

Create a program that uses tree recursion to print a pattern like the one shown below.

**Sample Run**

**Tree Pattern**

**Enter the number of branches: 5**

**1 \*\*\*\*\***

**2 \*\*\*\*\*\*\*\*\*\***

**1 \*\*\*\*\***

**3 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**1 \*\*\*\*\***

**2 \*\*\*\*\*\*\*\*\*\***

**1 \*\*\*\*\***

**4 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**1 \*\*\*\*\***

**2 \*\*\*\*\*\*\*\*\*\***

**1 \*\*\*\*\***

**3 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**1 \*\*\*\*\***

**2 \*\*\*\*\*\*\*\*\*\***

**1 \*\*\*\*\***

**5 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**1 \*\*\*\*\***

**2 \*\*\*\*\*\*\*\*\*\***

**1 \*\*\*\*\***

**3 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**1 \*\*\*\*\***

**2 \*\*\*\*\*\*\*\*\*\***

**1 \*\*\*\*\***

**4 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**1 \*\*\*\*\***

**2 \*\*\*\*\*\*\*\*\*\***

**1 \*\*\*\*\***

**3 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**1 \*\*\*\*\***

**2 \*\*\*\*\*\*\*\*\*\***

**1 \*\*\*\*\***

**Specifications**

 The program can only accept a positive number of branches in the tree. Since the

number of branches increases exponentially, this program will take a long time to

execute for numbers larger than 12 or so.

 Use the following recursive algorithm to generate the pattern shown above:

**if number = 0, end function**

**otherwise,**

**start branch for number - 1**

**print number and its visual representation**

**start branch for number - 1**

 To get the visual representation for a branch, you can multiply the asterisk (\*) by 5.

In other words, 1 is 5 asterisks, 2 is 10 asterisks, and so on.

**Project 15-2: Roshambo**

Create an object-oriented program for a Roshambo game where the user can choose to

compete against one of two computer players: Bart or Lisa.

**Sample Run**

**Roshambo Game**

**Enter your name: Joel**

**Would you like to play Bart or Lisa? (b/l): b**

**Rock, paper, or scissors? (r/p/s): r**

**Joel: rock**

**Bart: rock**

**Draw!**

**Play again? (y/n): y**

**Rock, paper, or scissors? (r/p/s): p**

**Joel: paper**

**Bart: rock**

**Joel wins!**

**Play again? (y/n): y**

**Rock, paper, or scissors? (r/p/s): s**

**Joel: scissors**

**Bart: rock**

**Bart wins!**

**Play again? (y/n): n**

**Thanks for playing!**

**Specifications**

 Create a class named Player that provides attributes for storing the player’s name and

Roshambo value.

 Create a class named Bart that inherits the Player class and adds a generateRoshambo

method. This method should set the Roshambo attribute to rock.

 Create a class named Lisa that inherits the Player class and adds a generateRoshambo

method. This method should randomly select rock, paper, or scissors.

 The program should allow the user (Player) to play Roshambo against Bart or Lisa.

 In the game of Roshambo, rock beats scissors, paper beats rock, and scissors beats

paper.

**Enhancement**

 Keep track of wins and losses and display them at the end of each session.