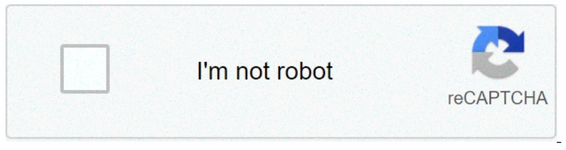
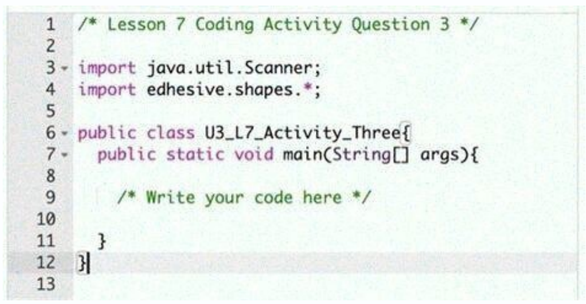


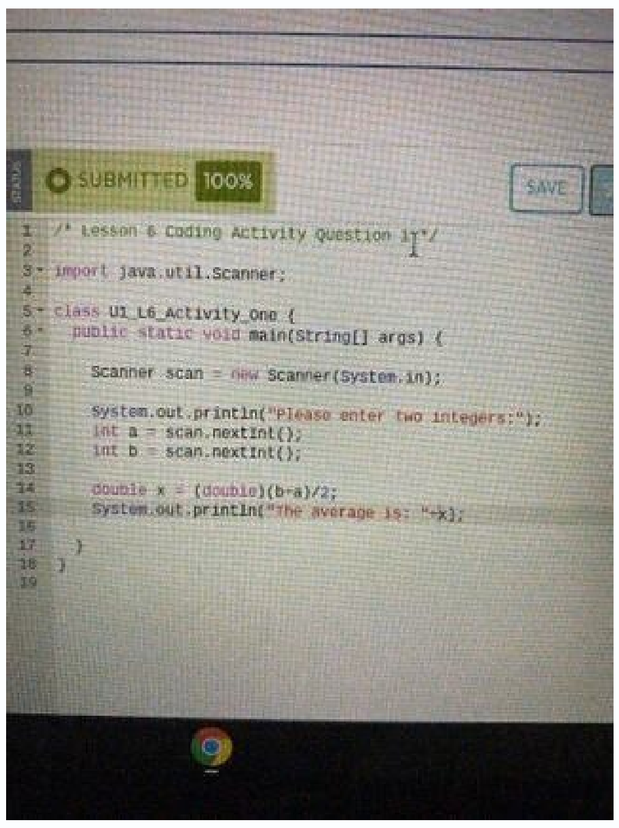
I am not robot!

Edhesive's lesson 4.2 provides an in-depth exploration of loop structures in programming, emphasizing their utility in repeating code blocks efficiently. The lesson includes practical exercises that reinforce the concept, such as input-driven loops that prompt users for information until a specific condition is met. For instance, a loop may continue to request pet names from the user until 'rock' is entered, incrementing a count with each iteration. The lesson also addresses common errors in loop implementation, such as the necessity of proper indentation to ensure commands are correctly executed within the loop's scope. Additionally, it presents various scenarios where loops output different results based on the initial conditions and incrementation patterns set by the programmer. In one exercise, students are tasked with adjusting a loop to print a sequence of numbers, demonstrating the control they have over the loop's execution. Another exercise involves a cumulative sum loop, where students must write code to calculate the sum of numbers entered by the user, showcasing the loop's ability to handle arithmetic operations and user input.

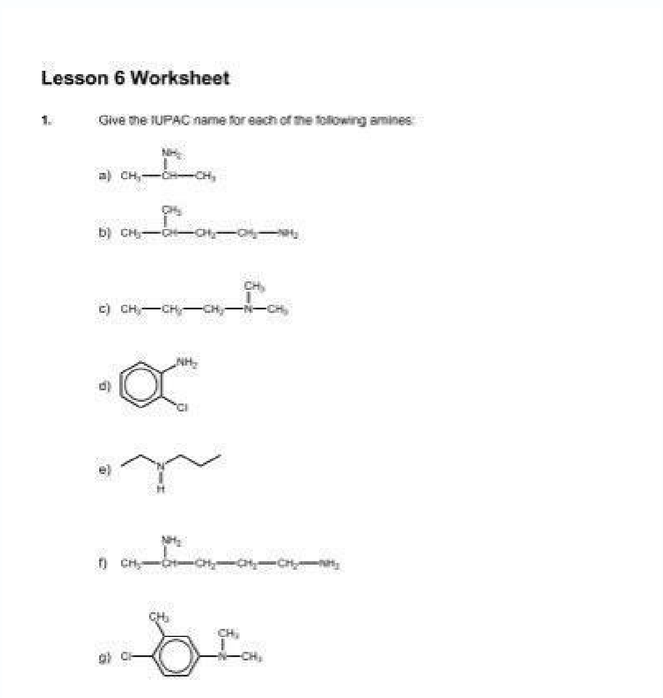


```
/* Lesson 7 Coding Activity Question 3 */

import java.util.Scanner;
import edhesive.shapes.*;

public class U3_L7_Activity_Three{
  public static void main(String[] args){

    /* Write your code here */

  }
}
```

In one exercise, students are tasked with adjusting a loop to print a sequence of numbers, demonstrating the control they have over the loop's execution. Another exercise involves a cumulative sum loop, where students must write code to calculate the sum of numbers entered by the user, showcasing the loop's ability to handle arithmetic operations and user input. The lesson concludes with a discussion on different types of loops, such as count-controlled loops, which terminate after a predetermined number of iterations. This foundational knowledge is crucial for students learning to code, as loops are a fundamental aspect of many programming tasks. By integrating these exercises into their study routine, learners can enhance their understanding of loops, a key element in programming that enables the creation of dynamic and efficient code.
The lesson aligns with the keyword '4.2 lesson practice edhesive,' ensuring that it is easily discoverable for those seeking resources on this topic. In this programming scenario, a user is prompted to input numbers until the number 17 is entered. The program calculates the sum of all even numbers entered before the termination number. This demonstrates a user input loop. In a variation where the user inputs a fixed number of entries, specifically 17, the loop becomes a count-controlled loop. For instance, consider a program that asks for the user's age and then prints "HUG" for each year of the user's age. This is an example of a loop that repeats a block of code a certain number of times based on the user's input. Another example is a loop starting with the number 3 and continuing until it reaches or exceeds 21, printing each multiple of 3. This illustrates the use of a while loop with a conditional statement. The Edhesive platform offers various lessons and practices, such as "4.2 Lesson Practice Edhesive," which focus on understanding and applying loops in programming. These exercises are designed to reinforce the concept of loops, which are fundamental in executing repeated tasks in code. For example, a loop that increments a counter from 0 to 10, adding 5 each time, will output the numbers 5 and 10. Similarly, a loop that starts at 3 and increments by 2 until it reaches 10 will output the numbers 5, 7, 9, and 11. In another exercise, a loop begins with a counter at 1 and a sum at 0. It increments the counter by 3 until it reaches 10, adding the new counter value to the sum each time, resulting in a final sum of 21. To understand the practical application of loops, consider a scenario where you need to enter a value that will make a loop print the numbers 60, 70, and 80 sequentially.
The correct increment value to achieve this would be 10.



Another exercise involves a cumulative sum loop, where students must write code to calculate the sum of numbers entered by the user, showcasing the loop's ability to handle arithmetic operations and user input. The lesson concludes with a discussion on different types of loops, such as count-controlled loops, which terminate after a predetermined number of iterations. This foundational knowledge is crucial for students learning to code, as loops are a fundamental aspect of many programming tasks. By integrating these exercises into their study routine, learners can enhance their understanding of loops, a key element in programming that enables the creation of dynamic and efficient code.
The lesson aligns with the keyword '4.2 lesson practice edhesive,' ensuring that it is easily discoverable for those seeking resources on this topic.
In this programming scenario, a user is prompted to input numbers until the number 17 is entered. The program calculates the sum of all even numbers entered before the termination number. This demonstrates a user input loop. In a variation where the user inputs a fixed number of entries, specifically 17, the loop becomes a count-controlled loop. For instance, consider a program that asks for the user's age and then prints "HUG" for each year of the user's age. This is an example of a loop that repeats a block of code a certain number of times based on the user's input. Another example is a loop starting with the number 3 and continuing until it reaches or exceeds 21, printing each multiple of 3.



In one exercise, students are tasked with adjusting a loop to print a sequence of numbers, demonstrating the control they have over the loop's execution. Another exercise involves a cumulative sum loop, where students must write code to calculate the sum of numbers entered by the user, showcasing the loop's ability to handle arithmetic operations and user input. The lesson concludes with a discussion on different types of loops, such as count-controlled loops, which terminate after a predetermined number of iterations. This foundational knowledge is crucial for students learning to code, as loops are a fundamental aspect of many programming tasks. By integrating these exercises into their study routine, learners can enhance their understanding of loops, a key element in programming that enables the creation of dynamic and efficient code. The lesson aligns with the keyword '4.2 lesson practice edhesive,' ensuring that it is easily discoverable for those seeking resources on this topic. In this programming scenario, a user is prompted to input numbers until the number 17 is entered. The program calculates the sum of all even numbers entered before the termination number. This demonstrates a user input loop. In a variation where the user inputs a fixed number of entries, specifically 17, the loop becomes a count-controlled loop.
For instance, consider a program that asks for the user's age and then prints "HUG" for each year of the user's age. This is an example of a loop that repeats a block of code a certain number of times based on the user's input. Another example is a loop starting with the number 3 and continuing until it reaches or exceeds 21, printing each multiple of 3. This illustrates the use of a while loop with a conditional statement. The Edhesive platform offers various lessons and practices, such as "4.2 Lesson Practice Edhesive," which focus on understanding and applying loops in programming. These exercises are designed to reinforce the concept of loops, which are fundamental in executing repeated tasks in code. For example, a loop that increments a counter from 0 to 10, adding 5 each time, will output the numbers 5 and 10. Similarly, a loop that starts at 3 and increments by 2 until it reaches 10 will output the numbers 5, 7, 9, and 11. In another exercise, a loop begins with a counter at 1 and a sum at 0. It increments the counter by 3 until it reaches 10, adding the new counter value to the sum each time, resulting in a final sum of 21. To understand the practical application of loops, consider a scenario where you need to enter a value that will make a loop print the numbers 60, 70, and 80 sequentially. The correct increment value to achieve this would be 10. In a code practice question, a loop is used to accumulate a sum of numbers entered by the user until the sum exceeds 100. The program also counts the number of entries made.
Lastly, consider a program that prompts for the type of pet owned. The loop continues until the user enters "rock," counting each entry made. These exercises from Edhesive, including the "4.2 Lesson Practice," are instrumental in teaching students the mechanics and applications of loops in programming. Understanding these concepts is essential for

developing efficient and effective code. In the context of the 4.2 lesson practice from Edhesive, the following code snippet is an example of how to manage user input and output in a Python program. The code demonstrates a simple interaction where the program counts the number of pets a user has and then prompts the user to enter the type of pet: ```python count = int(input("Enter the number of pets you have: ")) print("You have " + str(count) + " pet(s)") pet = input("Enter pet: ") ``` This exercise is part of the 4.2 lesson practice on Edhesive, which aims to teach students the basics of user interaction within a program. To access the full lesson and additional resources, you can download the Edhesive app or visit the app stores to find more information. This lesson is essential for understanding how to collect and process user input in Python, a fundamental skill in programming.