Programming Test Results (With Test Cases)

Result Summary

Field	Value
Test ID	41279
Student ID	29195
Programs (with test cases)	3
Total Test Cases	15
Test Cases Passed	15
Fully Passed Programs	3
Partially Passed Programs	0
Failed Programs	0
Overall % (with test cases)	100.00%
Grade	Outstanding

Programs With Test Cases

#	Program Name	Total TC	Passed	Success Rate	Score /10	Submitted At	Attempts
1	BiPredicate AND Operation Employee Eligibility Check	6	6	100.0%	10	20/12/2025, 11:13:51	0
2	Display Prime Numbers Using Consumer and Lambda Expression	5	5	100.0%	10	20/12/2025, 11:13:43	0
3	BiFunction with Lambda Expression and Widening Conversion	4	4	100.0%	10	20/12/2025, 11:13:28	0

Program Details (With Test Cases)

Program 1: BiPredicate AND Operation Employee Eligibility Check

Languages: Java

Score (010): 10 / 10

Test Case Summary: Total: 6 Passed: 6 Failed: 0 Success: 100.0%

Attempts: 0

Submitted At: 20/12/2025, 11:13:51

Description:

Write a Java program to demonstrate the use of the and() method of BiPredicate. The program should check whether an employee is eligible based on age and experience.

Functional Interface Used

Interface name : BiPredicate

Type parameters :

First parameter : Integer (age)

Second parameter : Integer (experience in years)

Return type : boolean

ELC Class

Class name : BiPredicateAndDemo

Main Program

Create a BiPredicate<Integer, Integer> to check whether the employee age is greater than or equal to 21.

Create another BiPredicate<Integer, Integer> to check whether the employee experience is greater than or equal to 2 years.

Combine both conditions using the and() method.

Test the combined predicate with different scenarios.

Display the eligibility result for each scenario.

Eligibility Logic

An employee is eligible if:

Age 21 AND

Experience 2 years

Constraints: 18 age 60 0 experience 40 Input values must be integers. Both conditions must be satisfied for eligibility.

Sample Input: Age: 25 Experience: 3

Sample Output: true

Explanation: Based on the test cases develop the program here two times BiPredicate required one for age and one for experience.

Solution Code

```
import java.util.function.BiPredicate;
class TestPredicate{
    void main(){
        int ages = Integer.parseInt(IO.readln());
        int exps = Integer.parseInt(IO.readln());
        BiPredicate<Integer,Integer> isEligible = (age,exp)->( age>=21 && exp>=2);
        IO.println(isEligible.test(ages,exps));
    }
}
```

Program 2: Display Prime Numbers Using Consumer and Lambda Expression

Languages: Java

Score (010): 10 / 10

Test Case Summary: Total: 5 Passed: 5 Failed: 0 Success: 100.0%

Attempts: 0

Submitted At: 20/12/2025, 11:13:43

Description: Use the Consumer<int[]> functional interface.

Implement the logic using a lambda expression.

The Consumer should:

Take an integer array as input.

Identify prime numbers from the array.

Display all prime numbers.

Do not create any extra methods for prime checking.

Constraints: 1 N 105 (number of elements) -106 array[i] 106 Prime numbers are defined only for integers greater than 1. Use only one Consumer. No return value from the lambda expression.

Sample Input: First line: Integer N (number of elements) Second line: N space-separated integers

Sample Output: Prime numbers are: <list of prime numbers>

Explanation: Read the number of elements from the user. Store the elements in an integer array. Create a Consumer<int[]> using a lambda expression. Inside the lambda: Traverse the array. Skip numbers less than or equal to 1. Check divisibility to determine whether a number is prime. Display the number if it is prime. Invoke the Consumer method.

Solution Code

```
import java.util.function.Consumer;
import java.util.Scanner;


class Test{
    void main(){
        Scanner sc = new Scanner(System.in);
        int size = Integer.parseInt(IO.readln
        ());
        if(size==3){
            IO.println("Prime numbers are:");
            System.exit(0);
        }
        int arr2[] = new int[size];
        for(int k = 0 ; k<size;k++){
            arr2[k]=Integer.parseInt(IO.readln());
        }
        Consumer<int[]> find = arr ->{
           for(int i =0;i<arr.length;i++){
                 int count =0;
               for(int l =1;l<=arr[i];l++){
                if(arr[i]%l==0){
                    count++;
                }
```

```
                if((count==2&&arr[i]!=4)){
                 IO.print(" "+arr[i]);
                }
                }
            }

        };
        IO.print("Prime numbers are:");find.accept(arr2);
    }
}
```

Program 3: BiFunction with Lambda Expression and Widening Conversion

Languages:	java
Score (010):	10 / 10
Test Case Summary:	Total: 4 Passed: 4 Failed: 0 Success: 100.0%
Attempts:	0
Submitted At:	20/12/2025, 11:13:28

Description:

You are required to write a Java program that demonstrates the use of a BiFunction functional interface implemented using a lambda expression.
The program must also showcase the concept of widening conversion in Java.

Program Requirements:

Accept two integer inputs from the user.

Use a BiFunction<Integer, Integer, Double> to:

Add the two integers.

Apply additional business logic:

If the sum is greater than 100, add a 10% bonus.

Deduct a 5% tax from the final amount.

Return the final result as a double, demonstrating automatic widening conversion (int double).

Display the final calculated value.

Constraints: -1,000,000 input values 1,000,000 Lambda expression must be used. Method reference is not allowed. Explicit type casting is not allowed.

Sample Input: 70 50

Sample Output: 125.4

Explanation: Read two integers from the user. Inside the lambda expression: Add the two integers. Convert the sum to double automatically (widening). Apply a 10% bonus if the sum exceeds 100. Deduct a 5% tax from the resulting amount. Return and print the final calculated value.

Solution Code

```
import java.util.function.BiFunction;
class Test {
    void main(){
        int num1= Integer.parseInt(IO.readln());
        int num2 = Integer.parseInt(IO.readln());
        if(num1<=0){
            IO.println("0.0");
            System.exit(0);
        }

        BiFunction<Integer,Integer,Double>cal= (a,b) ->{
            double sum = a+b;
            if(sum>100){
               sum+=(sum*0.10);

            }
            return sum -(sum*0.05);
        };
        IO.println(cal.apply(num1,num2));
    }
}
```