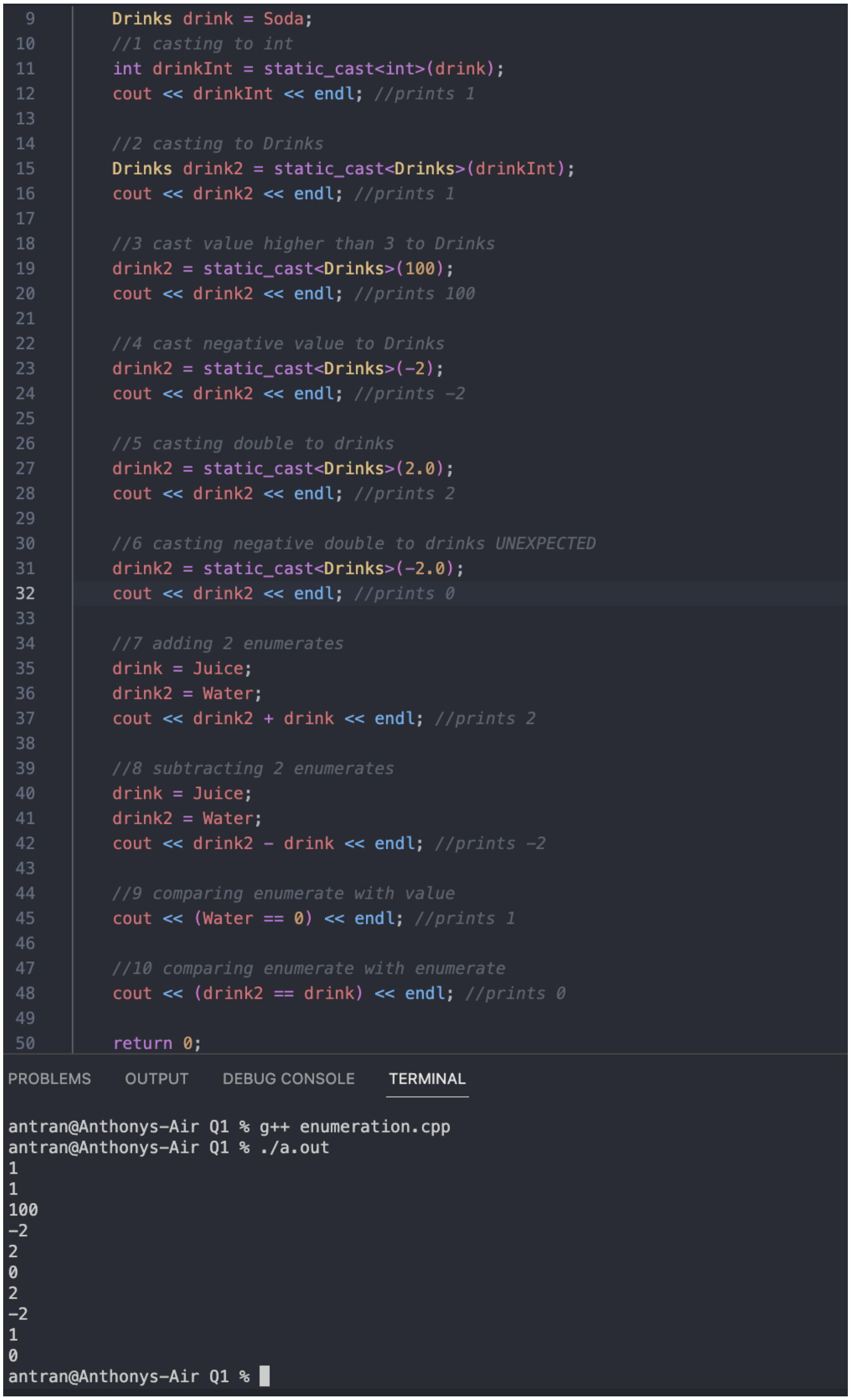
Anthony Tran
Axt220037
CS 4337-503

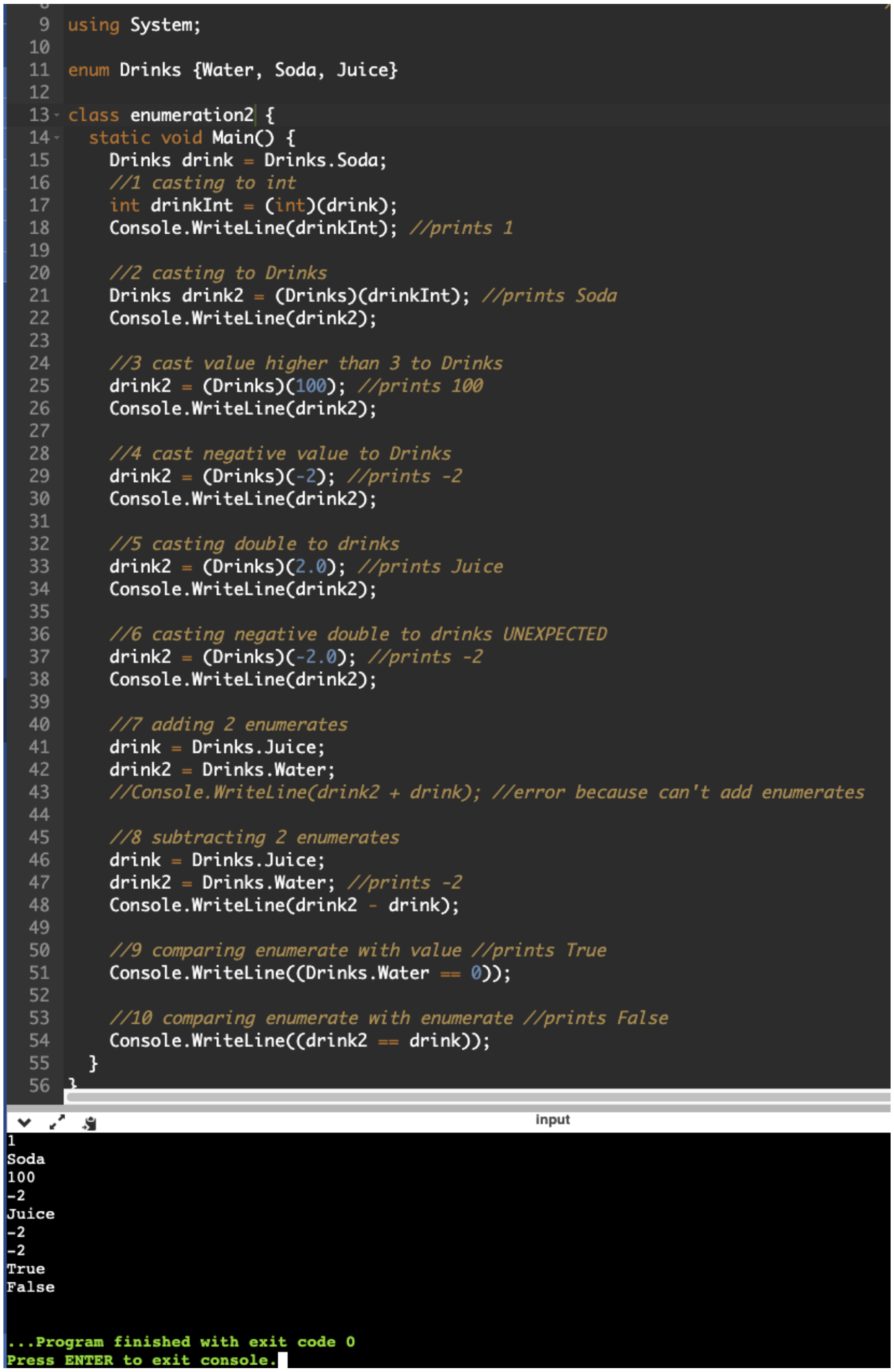
Q1-Q7 Screenshots

1. C++



```
    Drinks drink = Soda;
    //1 casting to int
    int drinkInt = static_cast<int>(drink);
    cout << drinkInt << endl; //prints 1

    //2 casting to Drinks
    Drinks drink2 = static_cast<Drinks>(drinkInt);
    cout << drink2 << endl; //prints 1

    //3 cast value higher than 3 to Drinks
    drink2 = static_cast<Drinks>(100);
    cout << drink2 << endl; //prints 100

    //4 cast negative value to Drinks
    drink2 = static_cast<Drinks>(-2);
    cout << drink2 << endl; //prints -2

    //5 casting double to drinks
    drink2 = static_cast<Drinks>(2.0);
    cout << drink2 << endl; //prints 2

    //6 casting negative double to drinks UNEXPECTED
    drink2 = static_cast<Drinks>(-2.0);
    cout << drink2 << endl; //prints 0

    //7 adding 2 enumerates
    drink = Juice;
    drink2 = Water;
    cout << drink2 + drink << endl; //prints 2

    //8 subtracting 2 enumerates
    drink = Juice;
    drink2 = Water;
    cout << drink2 - drink << endl; //prints -2

    //9 comparing enumerate with value
    cout << (Water == 0) << endl; //prints 1

    //10 comparing enumerate with enumerate
    cout << (drink2 == drink) << endl; //prints 0

    return 0;
```

```
antran@Anthonys-Air Q1 % g++ enumeration.cpp
antran@Anthonys-Air Q1 % ./a.out
1
1
100
-2
2
0
2
-2
1
0
antran@Anthonys-Air Q1 %
```



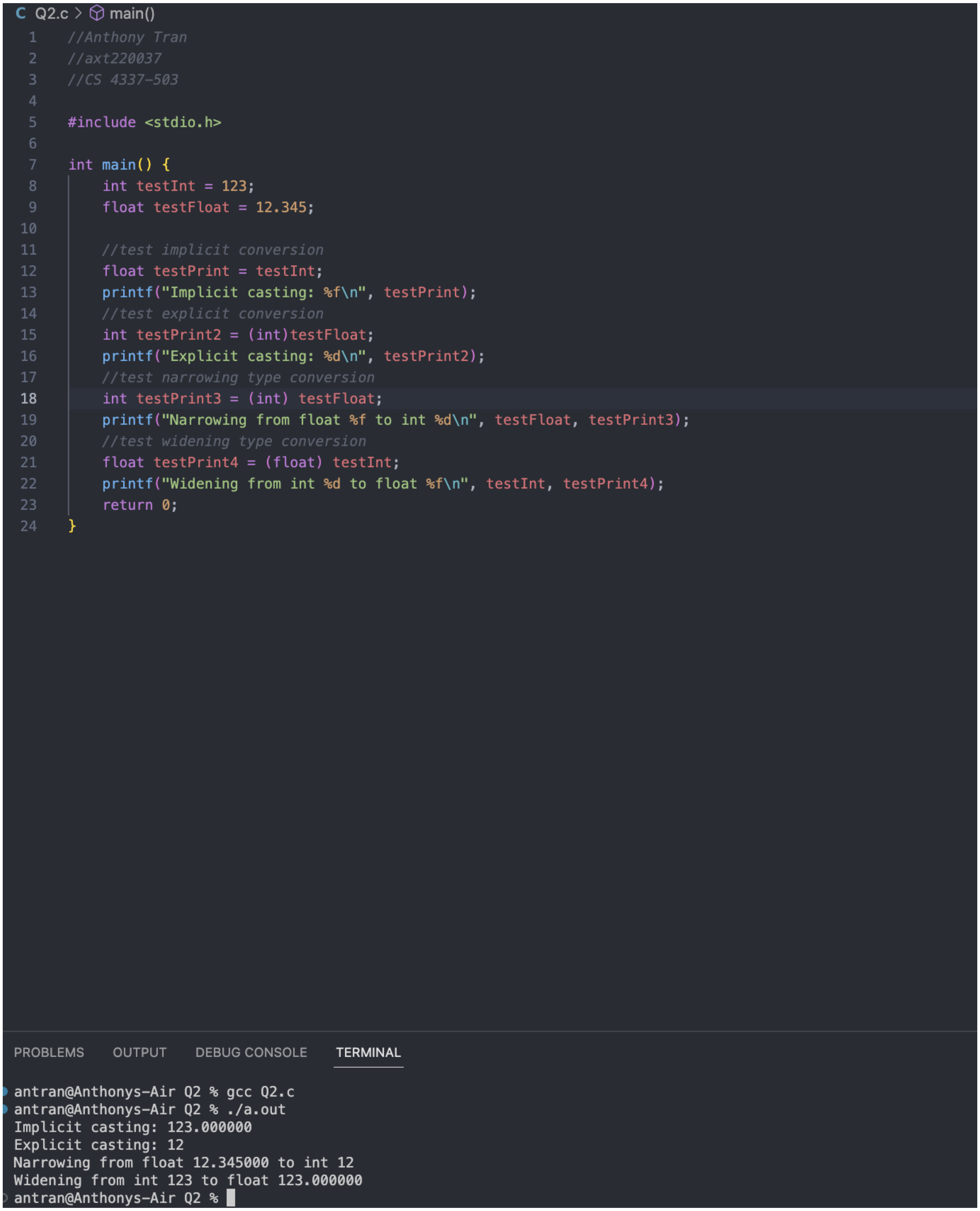
```
using System;

enum Drinks {Water, Soda, Juice}

class enumeration2 {
  static void Main() {
    Drinks drink = Drinks.Soda;
    //1 casting to int
    int drinkInt = (int)(drink);
    Console.WriteLine(drinkInt); //prints 1

    //2 casting to Drinks
    Drinks drink2 = (Drinks)(drinkInt); //prints Soda
    Console.WriteLine(drink2);

    //3 cast value higher than 3 to Drinks
    drink2 = (Drinks)(100); //prints 100
    Console.WriteLine(drink2);

    //4 cast negative value to Drinks
    drink2 = (Drinks)(-2); //prints -2
    Console.WriteLine(drink2);

    //5 casting double to drinks
    drink2 = (Drinks)(2.0); //prints Juice
    Console.WriteLine(drink2);

    //6 casting negative double to drinks UNEXPECTED
    drink2 = (Drinks)(-2.0); //prints -2
    Console.WriteLine(drink2);

    //7 adding 2 enumerates
    drink = Drinks.Juice;
    drink2 = Drinks.Water;
    //Console.WriteLine(drink2 + drink); //error because can't add enumerates

    //8 subtracting 2 enumerates
    drink = Drinks.Juice;
    drink2 = Drinks.Water; //prints -2
    Console.WriteLine(drink2 - drink);

    //9 comparing enumerate with value //prints True
    Console.WriteLine((Drinks.Water == 0));

    //10 comparing enumerate with enumerate //prints False
    Console.WriteLine((drink2 == drink));
  }
}
```

```
1
Soda
100
-2
Juice
-2
-2
True
False


...Program finished with exit code 0
Press ENTER to exit console.
```

C#

What I've noticed from these two comparisons is that C++ is more versatile and less restrictive than C#. This allows C# to be less vulnerable than C++ in that what you code is what you can expect. However, for C++, sometimes there are unexpected outcomes when it comes to casting. For C++, enumerates are printed and treated as ints so it is more versatile in having more operations. For C# enumerates bound-restricted and only those that are out of bounds will be printed as ints.

2. C compiler supports implicit, explicit, narrowing, and widening conversions.

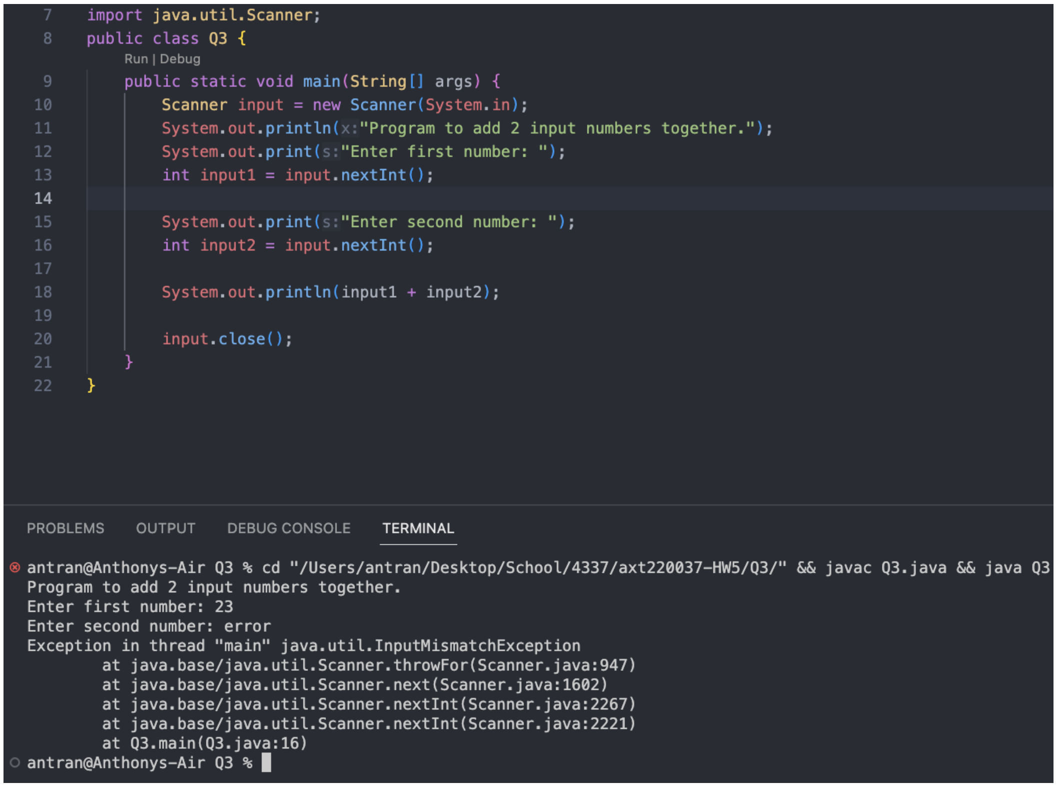


```c
//Anthony Tran
//axt220037
//CS 4337-503

#include <stdio.h>

int main() {
    int testInt = 123;
    float testFloat = 12.345;

    //test implicit conversion
    float testPrint = testInt;
    printf("Implicit casting: %f\n", testPrint);
    //test explicit conversion
    int testPrint2 = (int)testFloat;
    printf("Explicit casting: %d\n", testPrint2);
    //test narrowing type conversion
    int testPrint3 = (int) testFloat;
    printf("Narrowing from float %f to int %d\n", testFloat, testPrint3);
    //test widening type conversion
    float testPrint4 = (float) testInt;
    printf("Widening from int %d to float %f\n", testInt, testPrint4);
    return 0;
}
```

```
antran@Anthonys-Air Q2 % gcc Q2.c
antran@Anthonys-Air Q2 % ./a.out
Implicit casting: 123.000000
Explicit casting: 12
Narrowing from float 12.345000 to int 12
Widening from int 123 to float 123.000000
antran@Anthonys-Air Q2 %
```

3. CWE weaknesses screenshots

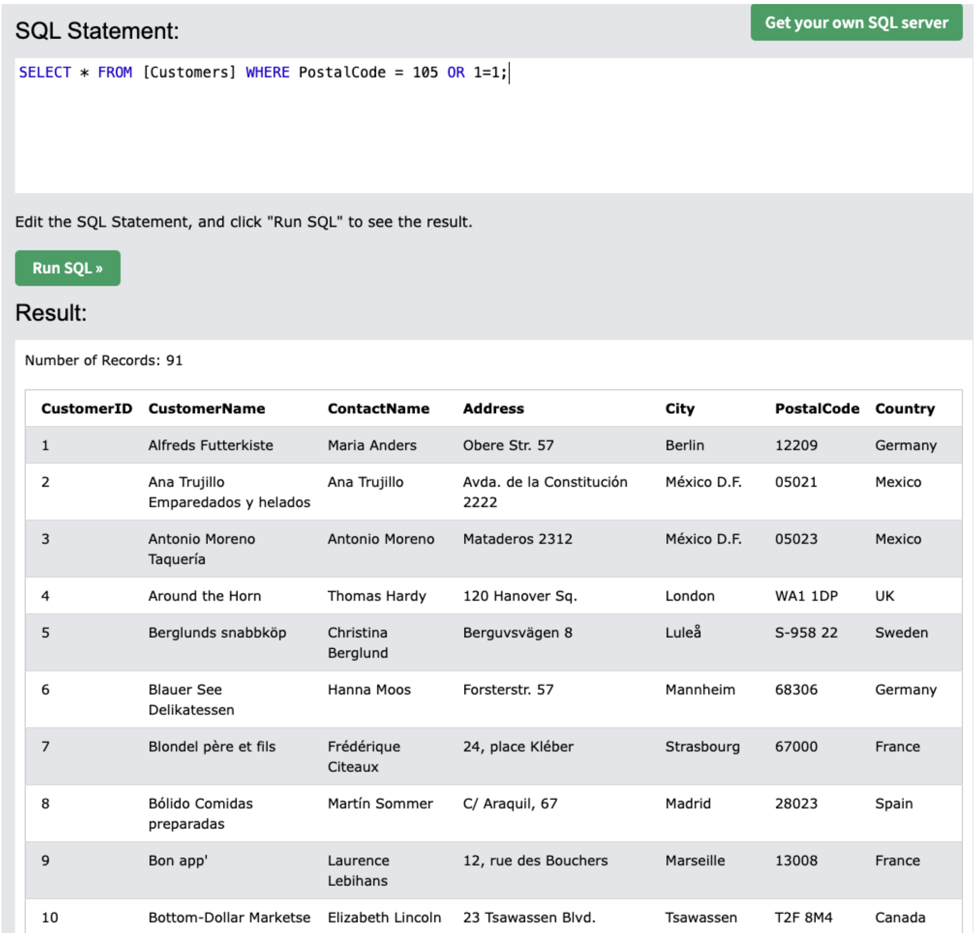
a. CWE-20 Improper Input Validation



```java
import java.util.Scanner;
public class Q3 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.println(x:"Program to add 2 input numbers together.");
        System.out.print(s:"Enter first number: ");
        int input1 = input.nextInt();

        System.out.print(s:"Enter second number: ");
        int input2 = input.nextInt();

        System.out.println(input1 + input2);

        input.close();
    }
}
```

```
⊗ antran@Anthonys-Air Q3 % cd "/Users/antran/Desktop/School/4337/axt220037-HW5/Q3/" && javac Q3.java && java Q3
  Program to add 2 input numbers together.
  Enter first number: 23
  Enter second number: error
  Exception in thread "main" java.util.InputMismatchException
          at java.base/java.util.Scanner.throwFor(Scanner.java:947)
          at java.base/java.util.Scanner.next(Scanner.java:1602)
          at java.base/java.util.Scanner.nextInt(Scanner.java:2267)
          at java.base/java.util.Scanner.nextInt(Scanner.java:2221)
          at Q3.main(Q3.java:16)
○ antran@Anthonys-Air Q3 % █
```

b. CWE-862 Missing Authorization. The entry of [PostalCode] OR 1=1 allows access to all customer information regardless of getting the PostalCode right, hence a sql injection.



SQL Statement:

```sql
SELECT * FROM [Customers] WHERE PostalCode = 105 OR 1=1;
```

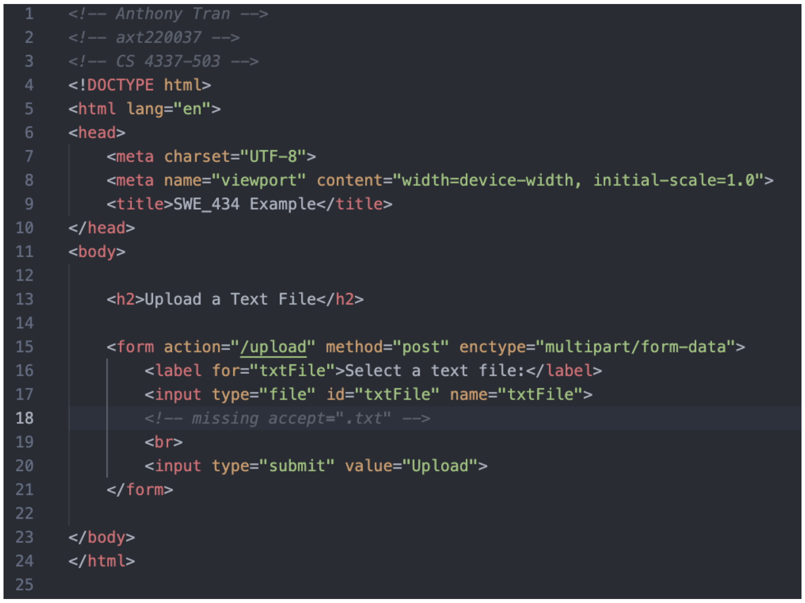
Edit the SQL Statement, and click "Run SQL" to see the result.

Result:

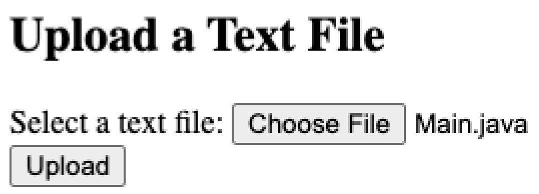
Number of Records: 91

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden
6	Blauer See Delikatessen	Hanna Moos	Forsterstr. 57	Mannheim	68306	Germany
7	Blondel père et fils	Frédérique Citeaux	24, place Kléber	Strasbourg	67000	France
8	Bólido Comidas preparadas	Martín Sommer	C/ Araquil, 67	Madrid	28023	Spain
9	Bon app'	Laurence Lebihans	12, rue des Bouchers	Marseille	13008	France
10	Bottom-Dollar Marketse	Elizabeth Lincoln	23 Tsawassen Blvd.	Tsawassen	T2F 8M4	Canada

c. CWE-434 Unrestricted upload of file with dangerous type. HTML code asks for upload of .txt code. However, the html tag doesn't have accept=".txt" so user can upload any file type, which can be hazardous.

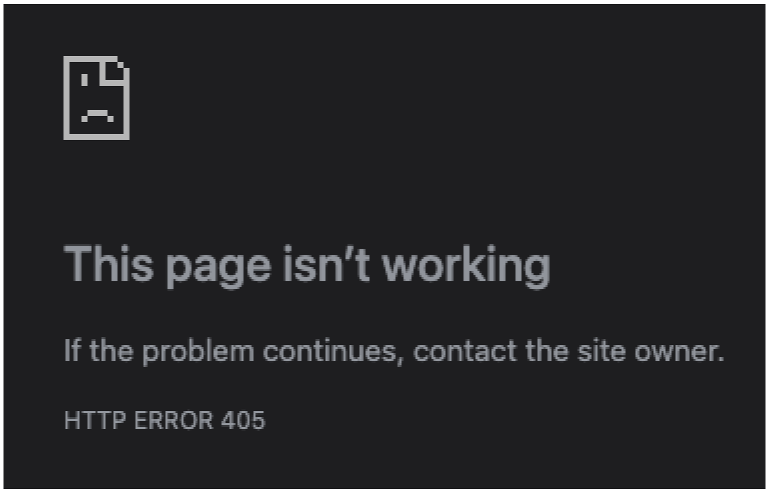


```html
<!-- Anthony Tran -->
<!-- axt220037 -->
<!-- CS 4337-503 -->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>SWE_434 Example</title>
</head>
<body>

    <h2>Upload a Text File</h2>

    <form action="/upload" method="post" enctype="multipart/form-data">
        <label for="txtFile">Select a text file:</label>
        <input type="file" id="txtFile" name="txtFile">
        <!-- missing accept=".txt" -->
        <br>
        <input type="submit" value="Upload">
    </form>

</body>
</html>

```



Upload a Text File

Select a text file: Main.java

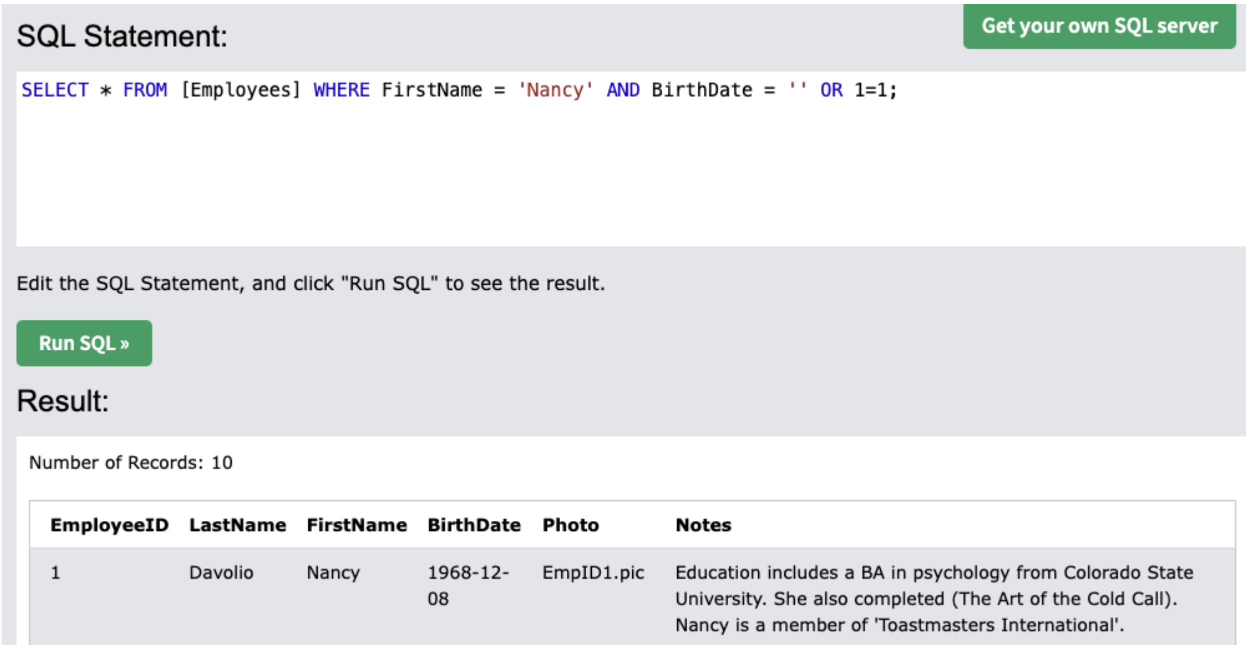


This page isn't working

If the problem continues, contact the site owner.

HTTP ERROR 405

d. CWE-89 Improper Neutralization of Special Elements. Similar to CWE-862 where user can enter ‘’ OR 1=1; to take advantage of sql quote function and have access to the database without having the necessary password.



SQL Statement:

```
SELECT * FROM [Employees] WHERE FirstName = 'Nancy' AND BirthDate = '' OR 1=1;
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Result:

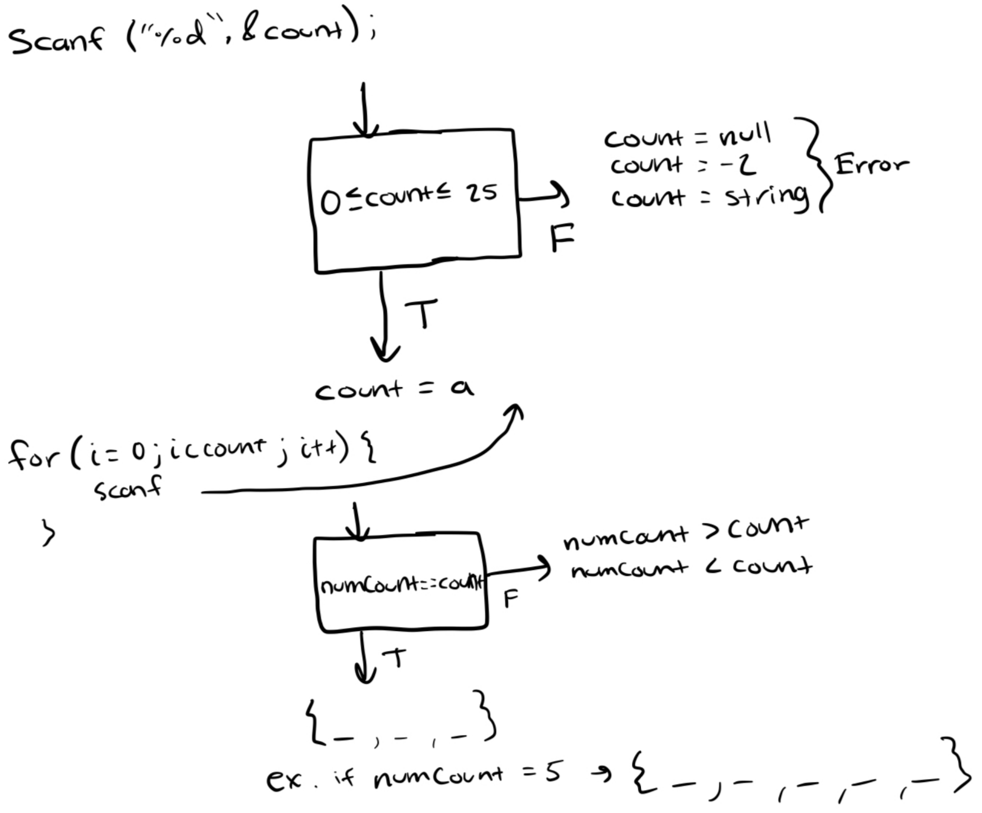
Number of Records: 10

EmployeeID	LastName	FirstName	BirthDate	Photo	Notes
1	Davolio	Nancy	1968-12-08	EmpID1.pic	Education includes a BA in psychology from Colorado State University. She also completed (The Art of the Cold Call). Nancy is a member of 'Toastmasters International'.

4. Cross-Site Request Forgery is when an attacker sends a request through the user's browser. This first works when the user has an active session in a website. The hacker then sends the user a malicious website where the website has hidden requests. These hidden requests may be in the form of html tags. Simultaneously, when the user visits the malicious website, the website sends a request to the targeted website. An example is "https://bank.com/change-password?newPassword=malicious," where the hacker can change the user's password. CSRF can be prevented by implementing CSRF tokens where each request is linked with a unique server-side generated token that is then shared with the user. In basic HTTP communication, each user connection is recognized by unique session tokens. For session hijacking, the hacker gains access to a user's session token by either stealing, predicting, or exploiting the token. Having the user's token, the hacker can now impersonate the user. To prevent session hijacking, using HTTPS can prevent data from being intercepted during transmission.
5.
    a. Black box testing makes sure the program is inline with description, hence we only look at the description. For the most part, we are checking if user input fits within array bounds, is not negative, is not null, is not a string, and cases for when input is 0.
    "Count for total number of elements, number[] to store the input in an array.
    Count = 0, expected = 0, actual 0
    Count = 5, expected = 5, actual 5
    Count = -2, expected = -2, actual -2
    Count = null, expected = error, actual error
    Count = string, expected = error, actual error
    "This loop would store the input numbers in array"
    Count = 0, expected = [], actual []

Count = 5, expected = [1,4,5,2,3], actual [1, 4, 5, 2, 3]
Count = 26, expected = error, actual error
Count = -2, expected = error, actual error
Count = null, expected = error, actual error
Count = string, expected = error, actual error
“Implementation of insertion sort algorithm”
Count = 0, expected = []
Count = 5, expected = [1,2,3,4,5]
Count = null, expected = error
Count = string, expected = error

b. White box testing



Count = 0, expected = [], actual []
Count = 5, expected = [1, 2, 3, 4, 5], actual [1, 2, 3, 4, 5]
Count = 26, expected = error, actual error
Count = -2, expected = error, actual error
Count = null, expected = error, actual error
Count = string, expected = error, actual error

c. Black box fuzzing enters random values to check for any unexpected behaviors.
Ex. (count = 5, 2, 4, hello, 4, 5; expected: error; actual: error)
(count = 3, 2, 3, 1; expected: [1, 2, 3]: actual: [1, 2, 3])

d. White box fuzzing uses random values but checks with the internal code.
Ex. (count = 5, 2, 3, 4, 1, 6, 2, 3: expected: error; actual: error)

User enters more numbers than count which results in a crash. Fuzzing allows programmers to catch test cases that would otherwise go unnoticed.

6. An XML file, also known as an eXtensible Markup Language, is a markup language that contains tags and attributes. To request and receive XML file, we can use HTTP get and post methods.

   Request (GET):

   ```
   GET /index.xml HTTP/1.1
   Host: example.com
   Content-Type: text/xml;
   charset=utf-8
   Content-Length: 123
   ```

   Response:

   ```
   HTTP/1.1 200 OK
   Date: Sun, 4 Dec 2023
   Server: Example
   <header>
   <body>
   ```

   The XML file is inside the body. Once the program requests XML through GET request, it sends a response when it receives the XML.

7. Type Conversions
   a. Ruby and PHP support Implicit Type Conversion. These languages support implicit conversion to make coding more convenient for the programmer. However, this comes at a cost where there may be unexpected behaviors.

      Ruby example: This code implicitly converts string to integer

      ```
      num = 1
      str = “2”
      result = num + str.to_i
      ```

      PHP example: This code implicitly converts string to integer

      ```
      $num = 1;
      $str = "2";
      $result = $num + $str;
      ```

   b. C# and Swift support Explicit Type Conversion. Explicit Type Conversion allows programmers to have more control and it helps avoid unexpected behaviors caused by implicit conversion.
      i. C# example: Explicit convert to double

         ```
         int num = 1;
         double num2 = (double)num;
         ```

         Swift example: Explicit convert to double

         ```
         let num = 1
         let num2 = Double(num)
         ```

   c. C and C++ support Narrowing Type Conversion. Narrow type conversion is used to translate larger to smaller types using truncation, this allows for more versatility between data types.

    i. C example: narrow conversion from double to int

        ```
        double num = 1.12;
        int num2 = (int) num;
        ```

        C++ example: narrow conversion from double to int

        ```
        double num = 1.12
        int num2 = static_cast<int> num;
        ```

d. Java and Python support Widening Type Conversion. Widening type conversion allows programmers convenience of converting smaller types to larger types without a loss of precision.
    i. Java example: widen conversion from int to double

        ```
        int num = 1;
        double num2 = num;
        ```

        Python example: widen conversion from int to float

        ```
        let num = 1
        let num2 = float(num)
        ```