CS 3250: Software Testing (Spring 2026)

POTD 5: Graph for source code (Data flow) – FizzBuzz – solution

Due 10-Mar-2026, 11:00am EST

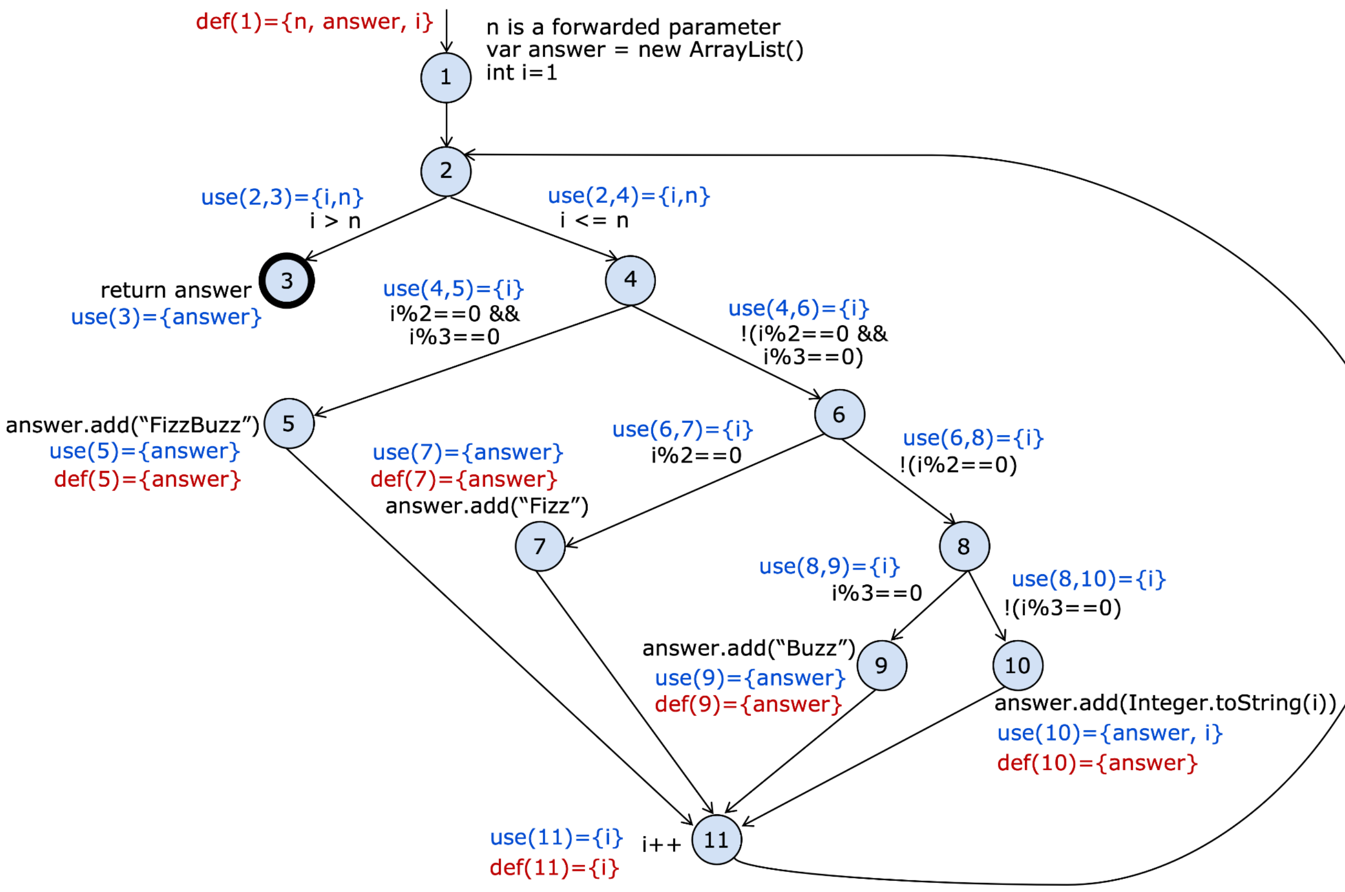
Purpose: Create graph representation for source code; apply data flow graph coverage to source code; get ready for assignment 4; prepare for quiz 3 and the final exam

You may make a copy of a worksheet and complete this activity, or type your answers in any text editor. You may work alone or with another student in this course (max team size=2).

Consider the following Java method

```java
public static List<String> fizzBuzz(int n)
{
   var answer = new ArrayList();
   for (int i = 1; i <= n; i++)
   {
      if (i % 2 == 0 && i % 3 == 0)
         answer.add("FizzBuzz");
      else if (i % 2 == 0)
         answer.add("Fizz");
      else if (i % 3 == 0)
         answer.add("Buzz");
      else
         answer.add(Integer.toString(i));
   }
   return answer;
}
```

1. Draw a **Control Flow Graph** for the fizzBuzz method. Annotate **all** information (i.e., source code, defs and uses).



2. List all **du-pairs**, then derive **du-paths** (the du-paths then can be used as test requirements)

   In reality, all variables must be analyzed and taken into account when designing tests.
   Due to the class time constraints, we will analyze and complete this part for a variable **answer** only.

   Note: the question asks for a variable answer only due to the class time constraints. This sample solution includes all variables.

DU-pairs	DU-paths
Variable **n**	
[1, (2,3)]	[1, 2, 3]
[1, (2,4)]	[1, 2, 4]
Variable **answer**	
[1, 3] *empty answer*	[1, 2, 3]
[1, 5]	~~[1, 2, 4, 5]~~ *semantically unreachable – i is initialized to 1. The first iteration results in [1,2,4,6,8,10] because i=1. Do not include [1,2,4,5,11,2,4,5] – no def-clear path.*
[1, 7]	~~[1, 2, 4, 6, 7]~~ *semantically unreachable – i is initialized to 1. The first iteration results in [1,2,4,6,8,10] because i=1.*
[1, 9]	~~[1, 2, 4, 6, 8, 9]~~ *semantically unreachable – i is initialized to 1. The first iteration results in [1,2,4,6,8,10] because i=1.*
[1, 10]	[1, 2, 4, 6, 8, 10]
[5, 3]	[5, 11, 2, 3]
[5, 5] *use before def*	~~[5, 11, 2, 4, 5]~~ *semantically unreachable – due to i++*
[5, 7]	~~[5, 11, 2, 4, 6, 7]~~ *semantically unreachable – due to i++*
[5, 9]	~~[5, 11, 2, 4, 6, 8, 9]~~ *semantically unreachable – due to i++*
[5, 10]	[5, 11, 2, 4, 6, 8, 10]
[7, 3]	[7, 11, 2, 3]
[7, 5]	~~[7, 11, 2, 4, 5]~~ *semantically unreachable – due to i++*
[7, 7] *use before def*	~~[7, 11, 2, 4, 6, 7]~~ *semantically unreachable – due to i++*
[7, 9]	[7, 11, 2, 4, 6, 8, 9]
[7, 10]	[7, 11, 2, 4, 6, 8, 10]
[9, 3]	[9, 11, 2, 3]
[9, 5]	~~[9, 11, 2, 4, 5]~~ *semantically unreachable – due to i++*
[9, 7]	[9, 11, 2, 4, 6, 7]
[9, 9] *use before def*	~~[9, 11, 2, 4, 6, 8, 9]~~ *semantically unreachable – due to i++*

[9, 10]	~~[9, 11, 2, 4, 6, 8, 10]~~ *semantically unreachable – due to i++*
[10, 3] [10, 5]	[10, 11, 2, 3] [10, 11, 2, 4, 5]
[10, 7]	[10, 11, 2, 4, 6, 7]
[10, 9]	~~[10, 11, 2, 4, 6, 8, 9]~~ *semantically unreachable – due to i++*
[10, 10] *use before def*	~~[10, 11, 2, 4, 6, 8, 10]~~ *semantically unreachable – due to i++*
Variable **i**	
[1, (2,3)]	[1, 2, 3]
[1, (2,4)]	[1, 2, 4]
[1, (4,5)]	~~[1, 2, 4, 5]~~ *semantically unreachable – i is initialized to 1. The first iteration results in [1,2,4,6,8,10] because i=1.* *Do not include [1,2,3,5,11,2,4,5] – no def-clear path.*
[1, (4,6)]	[1, 2, 4, 6]
[1, (6,7)]	~~[1, 2, 4, 6, 7]~~ *semantically unreachable – i is initialized to 1. The first iteration results in [1,2,4,6,8,10] because i=1.*
[1, (6,8)]	[1, 2, 4, 6, 8]
[1, (8,9)]	~~[1, 2, 4, 6, 8, 9]~~ *semantically unreachable – i is initialized to 1. The first iteration results in [1,2,4,6,8,10] because i=1.*
[1, (8,10)]	[1, 2, 4, 6, 8, 10]
[1, 10]	[1, 2, 4, 6, 8, 10]
[1, 11]	~~[1, 2, 4, 5, 11]~~ *semantically unreachable – i is initialized to 1.* ~~[1, 2, 4, 6, 7, 11]~~ *semantically unreachable – i is initialized to 1.* ~~[1, 2, 4, 6, 8, 9, 11]~~ *semantically unreachable – i is initialized to 1.* [1, 2, 4, 6, 8, 10, 11]
[11, (2,3)] [11, (2,4)] [11, (4,5)] [11, (4,6)] [11, (6,7)] [11, (6,8)] [11, (8,9)]	[11, 2, 3] [11, 2, 4] [11, 2, 4, 5] [11, 2, 4, 6] [11, 2, 4, 6, 7] [11, 2, 4, 6, 8] [11, 2, 4, 6, 8, 9]
[11, (8,10)] [11, 10]	[11, 2, 4, 6, 8, 10] [11, 2, 4, 6, 8, 10]
[11, 11] *use before def*	[11, 2, 4, 5, 11] [11, 2, 4, 6, 7, 11] [11, 2, 4, 6, 8, 9, 11] [11, 2, 4, 6, 8, 10, 11]

3. Apply **All-Defs** to design tests — some test paths tour multiple test requirements

   In reality, all variables must be analyzed and taken into account when designing tests.
   Due to the class time constraints, we will analyze and complete this part for a variable **answer** only.

   Note: the question asks for a variable answer only due to the class time constraints. This sample solution includes all variables.

variable	Test requirements	Test paths	Test cases (input values and expected output)
n	[1,2,3]	[1,2,3]	input: n=0, expected: answer=[]
answer	[1,2,3]	[1,2,3]	input: n=0, expected: answer=[]
	[5,11,2,3]	[1,2, 4,6,8,10,11, 2,4,6,7,11, 2,4,6,8,9,11, 2,4,6,7,11, 2,4,6,8,10,11, 2,4,5,11, 2,3]	input: n=6, expected: answer=[1,Fizz,Buzz,Fizz,5,FizzBuzz]
	[7,11,2,3]	[1,2, 4,6,8,10,11, 2,4,6,7,11, 2,3]	input: n=2, expected: answer=[1,Fizz]
	[9,11,2,3]	[1,2, 4,6,8,10,11, 2,4,6,7,11, 2,4,6,8,9,11, 2,3]	input: n=3, expected: answer=[1,Fizz,Buzz]
	[10,11,2,3]	[1,2, 4,6,8,10,11, 2,3]	input: n=1; expected: answer=[1]
i	[1,2,3]	[1,2,3]	input: n=0, expected: answer=[]
	[11, (2,3)]	[1,2, 4,6,8,10,11, 2,3]	input: n=1, expected: answer=[1]

4. Apply **All-Uses** to design tests

   In reality, all variables must be analyzed and taken into account when designing tests.
   Due to the class time constraints, we will analyze and complete this part for a variable **answer** only.

   Note: the question asks for a variable answer only due to the class time constraints. This sample solution includes all variables.

variable	Test requirements	Test paths	Test cases (input values and expected output)
n	[1,2,3] *For du [1, (2,3)]*	[1,2,3]	input: n=0, expected: answer=[]
	[1,2,4] *For du [1, (2,4)]*	[1,2, 4,6,8,10,11, 2,3]	input: n=1; expected: answer=[1]
answer	[1,2,3] *For du [1,3]* [1,2,4,6,8,10] *For du [1,10]*	[1,2,3] [1,2, 4,6,8,10,11, 2,3]	input: n=0, expected: answer=[] input: n=1; expected: answer=[1]

	[5,11,2,3] *For du [5,3]*	[1,2, 4,6,8,10,11, 2,4,6,7,11, 2,4,6,8,9,11, 2,4,6,7,11, 2,4,6,8,10,11, 2,4,5,11, 2,3]	input: n=6, expected: answer=[1,Fizz,Buzz,Fizz,5,FizzBuzz]
	[5,11,2,4,6,8,10] *For du [5,10]*	[1,2, 4,6,8,10,11, 2,4,6,7,11, 2,4,6,8,9,11, 2,4,6,7,11, 2,4,6,8,10,11, 2,4,5,11, 2,4,6,8,10,11, 2,3]	input: n=7, expected: answer=[1,Fizz,Buzz,Fizz,5,FizzBuzz,7]
	[7,11,2,3] *For du [7,3]*	[1,2, 4,6,8,10,11, 2,4,6,7,11, 2,3]	input: n=2; expected: answer=[1,Fizz]
	[7,11,2,4,6,8,9] *For du [7,9]*	[1,2, 4,6,8,10,11, 2,4,6,7,11, 2,4,6,8,9,11, 2,3]	input: n=3, expected: answer=[1,Fizz,Buzz]
	[7,11,2,4,6,8,10] *For du [7,10]*	[1,2, 4,6,8,10,11, 2,4,6,7,11, 2,4,6,8,9,11, 2,4,6,7,11, 2,4,6,8,10,11, 2,3]	input: n=5, expected: answer=[1,Fizz,Buzz,Fizz,5]
	[9,11,2,3] *For du [9,3]*	[1,2, 4,6,8,10,11, 2,4,6,7,11, 2,4,6,8,9,11, 2,3]	input: n=3, expected: answer=[1,Fizz,Buzz]
	[9,11,2,4,6,7] *For du [9,7]*	[1,2, 4,6,8,10,11, 2,4,6,7,11, 2,4,6,8,9,11, 2,4,6,7,11, 2,4,6,8,10,11, 2,3]	input: n=5, *(or n=4, we reuse test n=5)* expected: answer=[1,Fizz,Buzz,Fizz,5]
	[10,11,2,3] *For du [10,3]*	[1,2, 4,6,8,10,11, 2,3]	input: n=1; expected: answer=[1]
	[10,11,2,4,5] *For du [10,5]*	[1,2, 4,6,8,10,11, 2,4,6,7,11, 2,4,6,8,9,11, 2,4,6,7,11, 2,4,6,8,10,11, 2,4,5,11, 2,3]	input: n=6, expected: answer=[1,Fizz,Buzz,Fizz,5,FizzBuzz]
	[10,11,2,4,6,7] *For du [10,7]*	[1,2, 4,6,8,10,11, 2,4,6,7,11, 2,4,6,8,9,11, 2,4,6,7,11, 2,4,6,8,10,11, 2,4,5,11, 2,3]	input: n=6, *(or n=2, we reuse test n=6)* expected: answer=[1,Fizz,Buzz,Fizz,5,FizzBuzz]
i	[1,2,3] *For du [1,(2,3)]*	[1,2,3]	input: n=0, expected: answer=[]
	[1,2,4] *For du [1,(2,4)]*	[1,2, 4,6,8,10,11, 2,3]	input: n=1, expected: answer=[1]
	[1,2,4,6] *For du [1,(4,6)]*	[1,2, 4,6,8,10,11, 2,3]	input: n=1, expected: answer=[1]
	[1,2,4,6,8] *For du [1,(6,8)]*	[1,2, 4,6,8,10,11, 2,3]	input: n=1, expected: answer=[1]
	[1,2,4,6,8,10] *For du [1, (8,10)] and [1,10]*	[1,2, 4,6,8,10,11, 2,3]	input: n=1, expected: answer=[1]
	[1,2,4,6,8,10,11] *For du [1,11]*	[1,2, 4,6,8,10,11, 2,3]	input: n=1, expected: answer=[1]
	[11,2,3] *For du [11,(2,3)]*	[1,2, 4,6,8,10,11, 2,3]	input: n=1, expected: answer=[1]

	[11,2,4] *For du [11,(2,4)]*	[1,2, 4,6,8,10,11, 2,4,6,7,11, 2,4,6,8,9,11, 2,4,6,7,11, 2,4,6,8,10,11, 2,4,5,11, 2,3]	input: n=6, expected: answer=[1,Fizz,Buzz,Fizz,5,FizzBuzz]
	[11,2,4,5] *For du [11,(4,5)]*	[1,2, 4,6,8,10,11, 2,4,6,7,11, 2,4,6,8,9,11, 2,4,6,7,11, 2,4,6,8,10,11, 2,4,5,11, 2,3]	input: n=6, expected: answer=[1,Fizz,Buzz,Fizz,5,FizzBuzz]
	[11,2,4,6] *For du [11,(4,6)]*	[1,2, 4,6,8,10,11, 2,4,6,7,11, 2,4,6,8,9,11, 2,4,6,7,11, 2,4,6,8,10,11, 2,4,5,11, 2,3]	input: n=6, expected: answer=[1,Fizz,Buzz,Fizz,5,FizzBuzz]
	[11,2,4,6,7] *For du [11,(6,7)]*	[1,2, 4,6,8,10,11, 2,4,6,7,11, 2,4,6,8,9,11, 2,4,6,7,11, 2,4,6,8,10,11, 2,4,5,11, 2,3]	input: n=6, expected: answer=[1,Fizz,Buzz,Fizz,5,FizzBuzz]
	[11,2,4,6,8] *For du [11,(6,8)]*	[1,2, 4,6,8,10,11, 2,4,6,7,11, 2,4,6,8,9,11, 2,4,6,7,11, 2,4,6,8,10,11, 2,4,5,11, 2,3]	input: n=6, expected: answer=[1,Fizz,Buzz,Fizz,5,FizzBuzz]
	[11,2,4,6,8,9] *For du [11,(8,9)]*	[1,2, 4,6,8,10,11, 2,4,6,7,11, 2,4,6,8,9,11, 2,4,6,7,11, 2,4,6,8,10,11, 2,4,5,11, 2,3]	input: n=6, expected: answer=[1,Fizz,Buzz,Fizz,5,FizzBuzz]
	[11,2,4,6,8,10] *For du [11,(8,10)] and [11,10]*	[1,2, 4,6,8,10,11, 2,4,6,7,11, 2,4,6,8,9,11, 2,4,6,7,11, 2,4,6,8,10,11, 2,4,5,11, 2,3]	input: n=6, expected: answer=[1,Fizz,Buzz,Fizz,5,FizzBuzz]
	[11,2,4,6,8,10,11] *For du [11,11]*	[1,2, 4,6,8,10,11, 2,4,6,7,11, 2,4,6,8,9,11, 2,4,6,7,11, 2,4,6,8,10,11, 2,4,5,11, 2,3]	input: n=6, expected: answer=[1,Fizz,Buzz,Fizz,5,FizzBuzz]

5. Apply **All-DU-Paths** to design tests

   In reality, all variables must be analyzed and taken into account when designing tests.
   Due to the class time constraints, we will analyze and complete this part for a variable **answer** only.

   Note: the question asks for a variable answer only due to the class time constraints. This sample solution includes all variables.

variable	**Test requirements**	**Test paths**	**Test cases (input values and expected output)**
n	1,2,3] *For du [1, (2,3)]* [1,2,4] *For du [1, (2,4)]*	[1,2,3] [1,2, 4,6,8,10,11, 2,3]	input: n=0, expected: answer=[] input: n=1; expected: answer=[1]
answer	[1,2,3] *For du [1,3]* [1,2,4,6,8,10] *For du [1,10]* [5,11,2,3] *For du [5,3]*	[1,2,3] [1,2, 4,6,8,10,11, 2,3] [1,2, 4,6,8,10,11, 2,4,6,7,11, 2,4,6,8,9,11, 2,4,6,7,11, 2,4,6,8,10,11, 2,4,5,11, 2,3]	input: n=0, expected: answer=[] input: n=1; expected: answer=[1] input: n=6, expected: answer=[1,Fizz,Buzz,Fizz,5,FizzBuzz]

	[5,11,2,4,6,8,10] *For du [5,10]*	[1,2, 4,6,8,10,11, 2,4,6,7,11, 2,4,6,8,9,11, 2,4,6,7,11, 2,4,6,8,10,11, 2,4,5,11, 2,4,6,8,10,11, 2,3]	input: n=7, expected: answer=[1,Fizz,Buzz,Fizz,5,FizzBuzz,7]
	[7,11,2,3] *For du [7,3]*	[1,2, 4,6,8,10,11, 2,4,6,7,11, 2,3]	input: n=2; expected: answer=[1,Fizz]
	[7,11,2,4,6,8,9] *For du [7,9]*	[1,2, 4,6,8,10,11, 2,4,6,7,11, 2,4,6,8,9,11, 2,3]	input: n=3, expected: answer=[1,Fizz,Buzz]
	[7,11,2,4,6,8,10] *For du [7,10]*	[1,2, 4,6,8,10,11, 2,4,6,7,11, 2,4,6,8,9,11, 2,4,6,7,11, 2,4,6,8,10,11, 2,3]	input: n=5, expected: answer=[1,Fizz,Buzz,Fizz,5]
	[9,11,2,3] *For du [9,3]*	[1,2, 4,6,8,10,11, 2,4,6,7,11, 2,4,6,8,9,11, 2,3]	input: n=3, expected: answer=[1,Fizz,Buzz]
	[9,11,2,4,6,7] *For du [9,7]*	[1,2, 4,6,8,10,11, 2,4,6,7,11, 2,4,6,8,9,11, 2,4,6,7,11, 2,4,6,8,10,11, 2,3]	input: n=5, expected: answer=[1,Fizz,Buzz,Fizz,5]
	[10,11,2,3] *For du [10,3]*	[1,2, 4,6,8,10,11, 2,3]	input: n=1; expected: answer=[1]
	[10,11,2,4,5] *For du [10,5]*	[1,2, 4,6,8,10,11, 2,4,6,7,11, 2,4,6,8,9,11, 2,4,6,7,11, 2,4,6,8,10,11, 2,4,5,11, 2,3]	input: n=6, expected: answer=[1,Fizz,Buzz,Fizz,5,FizzBuzz]
	[10,11,2,4,6,7] *For du [10,7]*	[1,2, 4,6,8,10,11, 2,4,6,7,11, 2,4,6,8,9,11, 2,4,6,7,11, 2,4,6,8,10,11, 2,4,5,11, 2,3]	input: n=6, expected: answer=[1,Fizz,Buzz,Fizz,5,FizzBuzz]
i	[1,2,3] *For du [1,(2,3)]*	[1,2,3]	input: n=0, expected: answer=[]
	[1,2,4] *For du [1,(2,4)]*	[1,2, 4,6,8,10,11, 2,3]	input: n=1, expected: answer=[1]
	[1,2,4,6] *For du [1,(4,6)]*	[1,2, 4,6,8,10,11, 2,3]	input: n=1, expected: answer=[1]
	[1,2,4,6,8] *For du [1,(6,8)]*	[1,2, 4,6,8,10,11, 2,3]	input: n=1, expected: answer=[1]
	[1,2,4,6,8,10] *For du [1, (8,10)] and [1,10]*	[1,2, 4,6,8,10,11, 2,3]	input: n=1, expected: answer=[1]
	[1,2,4,6,8,10,11] *For du [1,11]*	[1,2, 4,6,8,10,11, 2,3]	input: n=1, expected: answer=[1]
	[11,2,3] *For du [11,(2,3)]*	[1,2, 4,6,8,10,11, 2,3]	input: n=1, expected: answer=[1]
	[11,2,4] *For du [11,(2,4)]*	[1,2, 4,6,8,10,11, 2,4,6,7,11, 2,4,6,8,9,11, 2,4,6,7,11, 2,4,6,8,10,11, 2,4,5,11, 2,3]	input: n=6, expected: answer=[1,Fizz,Buzz,Fizz,5,FizzBuzz]

	[11,2,4,5] *For du [11,(4,5)]*	[1,2, 4,6,8,10,11, 2,4,6,7,11, 2,4,6,8,9,11, 2,4,6,7,11, 2,4,6,8,10,11, 2,4,5,11, 2,3]	input: n=6, expected: answer=[1,Fizz,Buzz,Fizz,5,FizzBuzz]
	[11,2,4,6] *For du [11,(4,6)]*	[1,2, 4,6,8,10,11, 2,4,6,7,11, 2,4,6,8,9,11, 2,4,6,7,11, 2,4,6,8,10,11, 2,4,5,11, 2,3]	input: n=6, expected: answer=[1,Fizz,Buzz,Fizz,5,FizzBuzz]
	[11,2,4,6,7] *For du [11,(6,7)]*	[1,2, 4,6,8,10,11, 2,4,6,7,11, 2,4,6,8,9,11, 2,4,6,7,11, 2,4,6,8,10,11, 2,4,5,11, 2,3]	input: n=6, expected: answer=[1,Fizz,Buzz,Fizz,5,FizzBuzz]
	[11,2,4,6,8] *For du [11,(6,8)]*	[1,2, 4,6,8,10,11, 2,4,6,7,11, 2,4,6,8,9,11, 2,4,6,7,11, 2,4,6,8,10,11, 2,4,5,11, 2,3]	input: n=6, expected: answer=[1,Fizz,Buzz,Fizz,5,FizzBuzz]
	[11,2,4,6,8,9] *For du [11,(8,9)]*	[1,2, 4,6,8,10,11, 2,4,6,7,11, 2,4,6,8,9,11, 2,4,6,7,11, 2,4,6,8,10,11, 2,4,5,11, 2,3]	input: n=6, expected: answer=[1,Fizz,Buzz,Fizz,5,FizzBuzz]
	[11,2,4,6,8,10] *For du [11,(8,10)] and [11,10]*	[1,2, 4,6,8,10,11, 2,4,6,7,11, 2,4,6,8,9,11, 2,4,6,7,11, 2,4,6,8,10,11, 2,4,5,11, 2,3]	input: n=6, expected: answer=[1,Fizz,Buzz,Fizz,5,FizzBuzz]
	[11,2,4,5,11] *For du [11,11]*	[1,2, 4,6,8,10,11, 2,4,6,7,11, 2,4,6,8,9,11, 2,4,6,7,11, 2,4,6,8,10,11, 2,4,5,11, 2,3]	input: n=6, expected: answer=[1,Fizz,Buzz,Fizz,5,FizzBuzz]
	[11,2,4,6,7,11] *For du [11,11]*	[1,2, 4,6,8,10,11, 2,4,6,7,11, 2,4,6,8,9,11, 2,4,6,7,11, 2,4,6,8,10,11, 2,4,5,11, 2,3]	input: n=6, expected: answer=[1,Fizz,Buzz,Fizz,5,FizzBuzz]
	[11,2,4,6,8,9,11] *For du [11,11]*	[1,2, 4,6,8,10,11, 2,4,6,7,11, 2,4,6,8,9,11, 2,4,6,7,11, 2,4,6,8,10,11, 2,4,5,11, 2,3]	input: n=6, expected: answer=[1,Fizz,Buzz,Fizz,5,FizzBuzz]
	[11,2,4,6,8,10,11] *For du [11,11]*	[1,2, 4,6,8,10,11, 2,4,6,7,11, 2,4,6,8,9,11, 2,4,6,7,11, 2,4,6,8,10,11, 2,4,5,11, 2,3]	input: n=6, expected: answer=[1,Fizz,Buzz,Fizz,5,FizzBuzz]

Grading rubric

[Total: 10 points]: Done (or provide evidence of your attempt, full or reasonable effort)

- (5 points) — Providing evidence of your attempt, minimal effort

(-2.5 points) for 24 hours late (submitted after 10-Mar-2026 11am EST, by 11-Mar-2026 11am EST)
(-5 points) for 48 hours late (submitted after 11-Mar-2026 11am EST, by 12-Mar-2026 11am EST)

Submission

- Save your report as a .pdf file
- Upload your report (.pdf) to ***POTD 5 on Gradescope***.
- Connect your partner to your group on Gradescope so that everyone receives credit
- Each team submits only ***one*** copy

Making your submission available to instructor and course staff is ***your*** responsibility; if we cannot access or open your file, you will not get credit. Be sure to test access to your file before the due date.