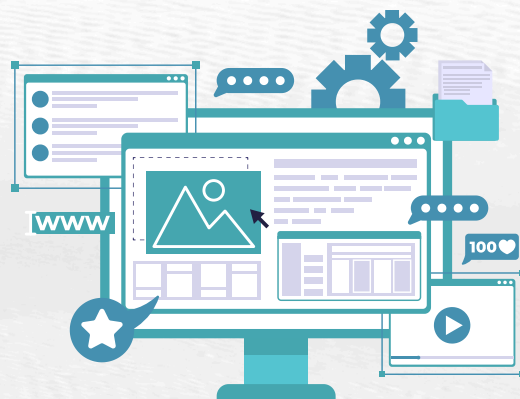


COMS 1270 TEST-OUT, FALL 2025

Thursday, August 21, 8:00a-10:00a
Wednesday, August 27, 7:00p-9:00p
B0029 Atanasoff



Guidance

COMS 1270 is one of the pre-requisites for COMS 2270. If at all possible you should take the test on Thursday, August 21 so that adjustments to your class schedule can be made before the semester starts on the 25th.

Students may attempt the test-out for COMS 1270 even if they have previously taken the course or 1270 test-out. Test-out results do not replace a grade received for a previously taken class. Test-out may be taken only once per semester.

Test-Out Registration

Register for the test-out examination at least one day before the exam date. Use this [Qualtrics link](#) to register.

Course Description for COMS 1270

Introduction to Computer Programming

Prereq: Credit or concurrent enrollment in MATH 1400 or higher

Introduction to computer programming with an emphasis on problem-solving. Topics include: program structures, expressions, variables, decision and logic, iteration, collections, input, and output. Program construction and testing. Programming assignments including games and applications. No prior programming experience necessary.

Procedures

You will be billed \$100 on your University bill a week after the test is completed, no matter the outcome-- failure to show up, failing the test, or passing the test.

The format of the test is pencil-and-paper. All of the problems will require writing actual, working code. The test will be graded by hand, and partial credit may be given for partially correct solutions.

The test is offered in Python or Java. However, if you know C, C++, Javascript, or another language, you can take the Python version and answer the questions as best you can in C, C++, Javascript, or another language.

You must bring your ISU student ID to be admitted to the test. Plan to arrive fifteen minutes before the start time so you can get checked in and be ready to begin on time. This is a timed exam and no late arrivals will be allowed.

Please check your ISU email two days in advance of the exam to verify location.

Sample Problems

Please see the practice problems included with this flyer to assess whether you might want to take the COMS 1270 test-out.

Visit the [Registrar's webpage](#) for more information about test-out at Iowa State.

Testing out of COMS 1270

COMS 1270 is the introductory programming course for students in Computer Science who have little or no previous programming experience. If you have already taken a high school or community college programming course, or if you have other prior experience, you should consider taking the test-out for COMS 1270 and starting with COMS 2270.

The format of the test is pencil-and-paper. All of the problems will involve writing actual code. The test will be graded by hand, and partial credit may be given for partially correct solutions. Some sample problems are given below.

The test will be offered in Python or in Java. If you know C, C++, Javascript, or another language, you can take the Python version and answer the questions as best you can using C, C++, Javascript, or another language.

The following sections provide detailed information about the test. There are two descriptions with sample problems, one for the Python version and one for the Java version.

Specific test information (Python version) and sample problems

Basic skills covered by the test:

- Declaring and using local variables
- Basic input and output
- Integer division and modulus operations
- Calling functions or methods (e.g. library functions, or functions defined in a different module, or methods of the string and list types)
- Defining functions according to a specification
- Conditional logic and Boolean operators
- For-loops
- While-loops
- Nested loops
- Strings, string operations
- Lists, list operations, creating lists

The test does NOT include:

- In-depth coverage of object-oriented programming beyond that required to use basic language features and utilities. E.g., in Python you need to understand how to *use* objects such as lists and strings, but you would not be expected to *define* your own classes.
- Programs requiring global data (file-scope or class-scope variables)
- Dictionaries or sets
- Reading/writing external files

Sample problems

1. Write a Python function `count_vowels` that, given a string, returns the number of vowels in the string (a character is defined to be a vowel if it is one of 'a', 'e', 'i', 'o', or 'u' or the uppercase equivalent). *For example, given string "Aardvark", the function returns 3.*

```
def count_vowels(s):  
    # TODO
```

2. A *Lucas sequence* is a sequence of integers in which each number (except the first two) is the sum of the previous two numbers in the list. The first two are arbitrary. Write a Python function `is_lucas_sequence` that takes a list of integers and returns True if the list is a Lucas sequence, and False otherwise. The function returns True for any sequence of two or fewer numbers. *For example: given the list [-7, 5, -2, 3, 1, 4, 5], the function should return True.*

```
def is_lucas_sequence(my_list):  
    # TODO
```

3. Donuts are \$1 each or \$10 for a dozen. Coffees are \$1.50 each, but you get a free coffee for each dozen donuts you buy. Write a Python function `coffee_break` that, given the desired number of donuts and coffees, returns the (best) price. *For example: for 33 donuts, the function returns \$29, since 33 is two dozen plus 9 single donuts. For 33 donuts and 4 coffees, the function returns 31.50: since a dozen costs only \$1 more than 9 singles, and that's less than the cost of a coffee, you'd buy three dozen and pay 1.50 more for the fourth coffee.*

```
def coffee_break(num_donuts, num_coffees):  
    # TODO
```

4. Suppose you are given a module named `foo` that includes the `coffee_break` function above. Write a user interface that obtains from the console the desired number of donuts and coffees,

and returns the price. A sample interaction should look like this, where the values in **bold** are entered by the user:

```
How many donuts? 33
How many coffees? 4
Your price is: 31.5
```

5. Suppose you are given a module named `foo` that includes a function `is_purple(n)`. What it does is: given any integer `n`, it returns `True` if the number is "purple" and `False` otherwise. We don't know what a "purple" number is, but we don't care, since this function is given to us! Write a Python function that, given an upper bound `max`, returns a list containing, in order, all the purple numbers between 0 and `max`, exclusive.

```
def find_purple_numbers(max):
    # TODO
```

6. Consider the Python function:

```
def mystery(x, y):
    result = False
    if x > y:
        if y != 0:
            result = True
    if x == 0:
        result = True
    return result
```

Notice that the function includes three conditional ("if") statements. Rewrite the function so that it produces exactly the same results, but does not include any conditional statements. (Partial credit may be given if you do it with just one "if" statement.)

7. Write a Python function `print_pattern` that, given any positive number `n`, produces `n` lines of output in the pattern illustrated below for the case `n = 5`:

```
5 4 3 2 1
4 3 2 1
3 2 1
2 1
1
```

```
def print_pattern(n):
    #TODO
```

Specific test information (Java version) and sample problems

Basic skills covered by the test:

- Declaring and using local variables
- Basic input and output; using Scanner for input
- Integer division and modulus operations
- Calling static or non-static methods (e.g. library methods, or those defined in a different class, or methods of the String and Scanner classes)
- Define static methods according to a specification
- Conditional logic and Boolean operators
- For-loops
- While-loops
- Nested loops
- Strings, string operations
- Arrays, creating arrays

The test does NOT include:

- In-depth coverage of object-oriented programming beyond that required to use basic language features and utilities. E.g., in Java you need to understand how to *construct and use* instances of Scanner and arrays, but you would not be expected to *define* your own classes.
- Programs requiring global data (static variables or instance variables)
- Maps, Sets, Lists
- Reading/writing external files

Sample problems

1. Write a static method `countVowels` that, given a string, returns the number of vowels in the string (a character is defined to be a vowel if it is one of 'a', 'e', 'i', 'o', or 'u' or the uppercase equivalent). *For example, given string "Aardvark", the method returns 3.*

```
class SomeClass {  
    public static int countVowels(String s) {  
        // TODO
```

2. A *Lucas sequence* is a sequence of integers in which each number (except the first two) is the sum of the previous two numbers. The first two are arbitrary. Write a static method that takes an array of integers and returns true if the array is a Lucas sequence, and false otherwise. The method returns true for any array of two or fewer numbers. *For example, given the array [-7, 5, -2, 3, 1, 4, 5], the method should return true.*

```
class SomeClass {
    public static boolean isLucasSequence(int[] arr) {
        // TODO
    }
}
```

3. Donuts are \$1 each or \$10 for a dozen. Coffees are \$1.50 each, but you get a free coffee for each dozen donuts. Write a static method `coffeeBreak` that, given the desired number of donuts and coffees, returns the (best) price. *For example: for 33 donuts, the function returns \$29, since 33 is two dozen plus 9 single donuts. For 33 donuts and 4 coffees, the function returns 31.50: since a dozen costs only \$1 more than 9 singles, and that's less than the cost of a coffee, you'd buy three dozen and pay 1.50 more for the fourth coffee.*

```
class SomeClass {
    public static double coffeeBreak(int numDonuts, int numCoffees) {
        // TODO
    }
}
```

4. Suppose you are given a class named `SomeClass`, located in package `somepackage`, that includes the `coffeeBreak` function as above. Write a user interface that obtains from the console the desired number of donuts and coffees, and returns the price. A sample interaction should look like this, where the values shown in **bold** represent responses typed by the user:

```
How many donuts? 34
How many coffees? 5
Your price is: 30.0
```

5. Suppose you are given a class named `Util`, located in package `somepackage` that includes a static method `isPurple(int n)`. What it does is: given any integer `n`, it returns true if the number is "purple" and false otherwise. We don't know what a "purple" number is, but we don't care, since this method is given to us! Write a static method that, given an upper bound `howMany`, returns an array containing, in order, the first `howMany` purple numbers. You can assume that all purple numbers are positive.

```
class SomeClass {
    public static int[] findPurpleNumbers(int howMany) {
        // TODO
    }
}
```

6. Consider the method:

```
public static boolean mystery(int x, int y)
{
    boolean result = false;
    if (x > y)
    {
        if (y != 0)
        {
            result = true;
        }
    }
    if (x == 0)
    {
        result = true;
    }
    return result;
}
```

Notice that the method includes three conditional ("if") statements. Rewrite the function so that it produces exactly the same results, but does not include any conditional statements. (Partial credit may be given if you do it with just one.)

7. Write a static method `printPattern` that, given any positive number `n`, produces `n` lines of output as shown below for the case `n = 5`:

```
5 4 3 2 1
4 3 2 1
3 2 1
2 1
1
```

```
class SomeClass {
    public static void printPattern(int n) {
        // TODO
    }
}
```