

## **CSC 111 – Introduction to Computer Science (Section C)**

*Course Description:* (4h) Lecture and laboratory. Rigorous introduction to the process of algorithmic problem solving and programming in a modern programming language.

Recommended as the first course for students considering a major or minor in computer science. Counts as a Division V divisional course.

*Lab:* One hour, fifteen minutes per week.

*Prerequisites:* Undergraduate status, cannot already have declared a CS major or minor; POI otherwise.

*Professor:* Dr. William Turkett - Manchester 240, 758-4427, email: turketwh@wfu.edu

*Office Hours:* (shared with CSC 112 class) MR 5:30-7:00pm, T 4:15-6:00pm, W 1:00-2:00pm, and by appointment

*Teaching Assistant:* Greg Galante (galagg8@wfu.edu), Harry Pham (phamtn8@wfu.edu), Shuai Zheng (zhens8@wfu.edu)

*Office Hours:* TBD

*Meeting Time: Lecture:* MWF 2:00 – 2:50pm, Manchester 241

*Lab:* Your lab section could be any of the following, depending on your schedule:

CSC 111 Lab A – Tuesday, 4:00-5:15pm, Dr. David John

CSC 111 Lab B – Thursday, 4:00-5:15pm, Dr. William Turkett

CSC 111 Lab C – Monday, 3:00-4:15pm, Dr. David John

Even though there are multiple sections of the class, please do not attend any session except for the one you are registered for, as there are space limitations in the room the course is taught in.

*Webpage:* <http://turkett111.tumblr.com>

*Textbook:* **A Gentle Introduction to Concurrent Programming**, by Wittman, Mathur, and Korb, available only at the Wake Forest University Bookstore.

*Grading:*

- 3 Tests - 40%
- 13 Lab Assignments - 20%
- Final Exam - 40%

Gaining hands-on experience with problem solving and programming is fundamental to learning Computer Science, so it is a policy for this course that a failing lab average will automatically result in a failing grade for the entire course.

*Expected Grading Scale:*

- |      |        |   |       |    |       |
|------|--------|---|-------|----|-------|
| • A+ | 97-100 | A | 92-96 | A- | 90-91 |
| • B+ | 87-89  | B | 82-86 | B- | 80-81 |
| • C+ | 77-79  | C | 72-76 | C- | 70-71 |
| • D+ | 67-69  | D | 62-66 | D- | 60-61 |
| • F  | 0-59   |   |       |    |       |

*Attendance:*

Regular attendance in class and at labs is expected.

*Tests and Final Exam:*

There will be three tests during the semester to judge the student's progress in the course. These tests may include material from any readings, lectures, and labs. The final will be cumulative over the material for the entire semester. All tests and exams will be closed book. Make up tests will be allowed only if the absence is excused by the University.

*Programming Lab:*

Programming Labs begins the week starting August 31st and will be held in Manchester 241. Please bring your laptop, textbook, course notes, lab assignment, and any pre-lab materials to each lab.

*Academic Integrity:* All work outside of the lab session should be done independently by each student. Copying of partial or complete work will be referred to the University Judicial System. You should keep evidence when possible to demonstrate your own work. Should a question of authorship arise you will be expected to produce documents that trace the development of your work. Algorithmic and electronic means of detecting copying may be used by the instructor on submitted assignments.

*Learning Assistance:* If you have a disability that may require an accommodation for taking this course, please contact the Learning Assistance Center (758-5929) within the first two weeks of the semester.

*Course Calendar:*

Wednesday, August 26 – First day of class

Wednesday, September 30 – 1st of three tests, Last day to drop with a W

Friday, October 16 – Fall Break holiday

Sunday, October 18 – Midterm grades posted

Monday, October 26 – 2nd of three tests

Friday, November 20<sup>th</sup> – 3rd of three tests

Wednesday, November 25<sup>th</sup> – Sunday, November 29<sup>th</sup> – Thanksgiving holiday

Friday, December 4 – Last day of class

Friday, December 11 – Final exam, 9:00am

*University Closure:* In the event that the University closes due to pandemic or other disaster, you will be provided with my home address, phone number, and a *CSC 111 Lecture Plan* document. You are requested to read the textbook material denoted within that document. Lecture and lab materials, in the form of Powerpoint slides and/or videos; lab programming exercises; and examination materials will be distributed electronically via the web, email, or via postal mail during the closure period. If the Internet is available, you should send electronic versions of your answers to the assignments to either my WFU email address or [turketwh@gmail.com](mailto:turketwh@gmail.com). Tests should be taken closed book, without access to papers, persons, or other resources, and submitted via postal mail. A return date for the examinations will be specified in the mailing.

*Topics Covered:*

- Logical problem solving
- Relationship between problem solving and computer programming
- Fundamental elements of modern computer programming
  - Variables and Types
  - Operators
  - Control structures
    - Conditionals
    - Loops
  - Collections of data
    - Arrays
    - Data structures
  - Methods
  - Object oriented programming
    - Classes and objects
    - Inheritance and polymorphism
  - Input and output, interfaces
  - Concurrency
- Java programming language
- Software development environments

*Course Outcomes:*

By the end of this course, students will:

*Algorithms & Problem Solving*

- Understand the differences between the concepts of a problem, an algorithm, and an implementation.
- Understand at a basic level the notion of concurrency, with respect to both computer architecture (multiple processors) and software (multiple tasks).
- Design an algorithm for a simple problem, independent of implementation
- Design an algorithm for a simple problem exploiting concurrency, independent of implementation

### *Programming Language Fundamentals*

- Understand the definition of the general meaning of variables and variable types as used in computer programming languages
- Be able to explain the differences between each of the primitive types commonly found in programming languages and between primitive types and user-defined types
- Understand the concept of container (aggregate) data-structures, with particular depth of knowledge in arrays
- Be able to define the fundamental selection and repetition structures of modern programming languages and to choose structures appropriate for different problem scenarios
- Understand at an intermediate level the concept of procedural (function-based) programs and at an elementary level the concepts underlying OOP (classes and objects) and demonstrate an ability to decide between these approaches appropriately for a given problem.
- Understand the basic notions of command line and graphical user interfaces

### *Implementations & Java*

- Be able to read and explain how intermediate level programs, written in the Java programming language, work in solving a problem
- Demonstrate an ability to correctly modify simple procedural and OOP programs written in the Java programming language to obtain new outcomes.
- Demonstrate an ability to design and implement from scratch, simple procedural programs employing primitive types, container types, selection and repetition structures, input/output primitives, and interface components using the Java programming language
- Demonstrate an ability to design and implement, from scratch, simple programs employing basic OOP concepts using the Java programming language
- Demonstrate an ability to design and implement, from scratch, simple programs employing basic concurrency concepts using the Java programming language
- Understand the definition of external libraries and APIs and to employ external libraries within a program in the Java language.

### *Software Development Processes*

- Understand the role of compilation and the difference between high-level programming languages and how programs are executed on an underlying architecture
- Understand the difference between syntactical and logical program errors and be able to find and repair such errors
- Be able to employ, and explain the purpose of, simple unit testing during program development
- Demonstrate the ability to work within a simple integrated software development environment.
- Demonstrate the ability to document developed code at a basic level.

*Planned Schedule:*

<b>Topic/Book Chapter</b>	<b>Date</b>
Chapter 1 – Introduction, Problem solving and programming	8/26 → 9/7
Chapter 2 – Variables and types, Operators	9/9 → 9/18
Chapter 3 – Control flow	9/21 → 10/2 (Test 1)
Chapter 4 – Constructing interfaces	10/5 → 10/14
Chapter 5 – Data structures and arrays	10/19 → 10/26 (Test 2)
Chapter 6 – Methods and classes	10/30 → 11/11
Chapter 8 – Concurrent programming	11/13 → 11/30 (Test 3)

The above schedule is tentative and is likely to be updated as the course proceeds.