

CSC 111B – Introduction to Computer Science (Section A): Processing

Course Description: (4h) Lecture and laboratory. Introduction to the basic concepts of computer programming and algorithmic problem solving for students with little or no programming experience. Recommended as the first course for students considering a major or minor in computer science; also appropriate for students who want computing experience applicable to other disciplines. Counts as a Division V divisional course. The 'B' version of the class is focused on multimedia and game computing.

Lab: One hour, fifteen minutes meeting 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 101 lab students) 1:00-1:50pm MWF, 1:00-3:00pm R, drop-in (if door open), by appointment

Teaching Assistant: Ms. Tess Stamper (stamte0@wfu.edu), Mr. Jacob White (whitj210@wfu.edu)

Office Hours: TBA in Manchester 242

Meeting Time: Lecture: MWF 12:00 – 12:50pm, Manchester 241

Lab: Thursday, 3:00-4:15, Manchester 241

Note: Even though there are multiple sections of the CSC111 class, each section is focused on a different programming language, so it is not possible to attend alternative lab sections as the material covered will not be the same.

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

Textbook: **Learning Processing**, by Daniel Shiffman, available at the Wake Forest University Bookstore or on the Internet, such as at [Amazon](#). You are also responsible for purchasing an [iClicker clicker](#). These are available at the bookstore and should be bought back at the end of the semester similar to textbooks. This clicker is the same one used by the WFU Physics, Chemistry, and Biology Departments, so you can reuse yours if you already have one.

Grading:

- Two Tests – 15% Each (30% Total)
- Weekly Quizzes (Fridays) – 10% Total
- Homeworks -25% Total
- Lab Assignments – 15% Total
- Final Exam – 20%

Expected Grading Scale:

•		A	93-100	A-	90-92	
•	B+	87-89	B	83-86	B-	80-82
•	C+	77-79	C	73-76	C-	70-72
•	D+	67-69	D	63-66	D-	60-62
•	F	0-59				

Attendance:

Regular attendance in class and at labs is expected. If you participate in an activity with excused absences (such as varsity sports), please provide documentation for those absences during the first week of class.

You should bring your laptop to all lecture and lab sessions. We will occasionally make use of the laptop during lectures, and will always make use of the laptop during the lab sessions. However, during lectures, I encourage you not to make use of the laptop unless you are actively taking notes for this class on the laptop or if you performing an activity for the class.

Tests and Final Exam:

There will be two tests during the semester to judge the student's progress in the course, as well as a final exam. These tests may include material from any readings, lectures, homeworks, and labs. The final will be cumulative over the material for the entire semester, but biased towards the material from the last third of the course. All tests and exams will be closed book. Make up tests will be allowed only if the absence is excused by the University. Please let me know before the test if you anticipate missing.

Programming Lab:

Programming Labs begins the first week of class (August 26th) and will be held in Manchester 241. Please bring your laptop, textbook, and course notes to the lab. Lab documents will be posted online before the lab – please print them at home if you desire a hard-copy.

Academic Integrity: All work performed outside of the in-class lecture and lab sessions should be done independently by each student unless otherwise specified. 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 25 – First day of class

Thursday, August 26 – First day of lab

Friday, September 24 – Test 1

Wednesday, September 29 – Last day to drop with a ‘W’

Friday, October 15 – Fall Break holiday

Sunday, October 20 – Midterm grades posted

Friday, October 29 – Test 2

Wednesday, November 24th – Sunday, November 28th – Thanksgiving holiday

Friday, December 3 – Last day of class

Friday, December 10 – Final exam, 2:00pm

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 as noted 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 or 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. Office hours will be held under such circumstances using instant messenger and video tools, with contact information for such tools provided upon university closure. Tests should be taken closed book, without access to papers, persons, or other resources, and submitted via postal mail. A return date for any such 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
 - Input and output, interfaces
- Interaction driven programming
- Digital media principles
 - Key digital media concepts
 - Programming with digital media
- The *Processing* and *Java* programming languages
- 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.
- Design an algorithm for a simple problem, independent of implementation
- Be able to identify the properties of good algorithms and apply this ability to critique an algorithm for a simple problem
- Be able to recognize and explain simple algorithms that are defined recursively

Programming Language Fundamentals

- Understand the definition of the general meaning of variables and variable types as used in computer programming languages and the role of assignment
- 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, and appropriate applications of these data-structures

- 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.
- Be able to decompose a given simple program into smaller pieces using the techniques of functional decomposition
- Understand the basic notions behind command line interfaces, graphical user interfaces, and file input/output

Implementations, Processing, and Java

- Be able to read and explain how intermediate level programs, written in the Processing programming language, work in solving a problem
- Be able to read and explain how simple 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 Processing and Java programming languages 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 Processing and Java programming languages
- Demonstrate an ability to design and implement, from scratch, simple programs employing basic OOP concepts using the Processing and Java programming languages
- Understand the definition of external libraries and APIs and to employ external libraries within a program in the Processing and Java languages.

Software Development Processes

- Understand the roles of compilation and interpretation 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.