PSU Expectations of Student Competencies

CS162: Introduction to Computer Science

The goals of CS162 are to teach the fundamental syntax of C++ to students who already know how to program and to prepare students to take CS163, Data Structures. Students coming into CS162 are expected to already be able to write complete programs in a high level programming language prior to taking CS162. This pre-requisite includes concepts such as: data types, variables, conditionals, loops, functions, arguments, and arrays. CS162 maps these concepts to C++ and then covers the use of structures, external data files, classes, pointers, dynamic memory, linear linked lists, recursion, and multi-dimensional arrays. Students are expected to become proficient programming using pointers, dynamic memory, and linear linked lists; students are expected to be introduced to the concepts of recursion, pointer arithmetic and multi-dimensional arrays to prepare for CS163, Data Structures.

After taking CS162, students should be able to write complete programs in C++, using multiple files (.cpp and .h files) implementing pointer based linear linked list solutions. Students should be competent using the linux environment without the assistance of an IDE, the internet, or external library support (beyond iostream, cctype, string and/or cstring libraries). Students should also be fluent with the use of arguments (both pass by value and pass by reference) rather than relying on global variables. Good use of structured programming techniques is highly recommend to avoid common programming pitfalls.

From a computer science perspective, students are expected to become proficient at writing algorithms, describe the ethical implications of software being developed, and experience developing and using test plans.

Students taking PSU's CS162 course implement 5 individual programming assignments, which include at least one program using external data files, two programs using classes and dynamically allocated arrays, and one program implementing linear linked lists from scratch. Students taking PSU's CS162 course also experience 10 labs during the term that provide hands on practice with linux, external data files, classes, linear linked list algorithms, and recursion. All work is done using linux; no IDEs are allowed.

PSU recommends the use of proficiency demos to determine the overall skill level of the students taking CS162 and recommends passing only students who show fluency with **pointer based linear linked list** solutions. Students passing CS162 demonstrate fluency at problems such as (for example):

- Adding a node at the end of a linear linked list
- Counting the number of nodes in a linear linked list
- Finding how many times a matching item appears in a linear linked list
- Removing an item from a linear linked list

When examining fluency, students should be able to develop code that is free from segmentation faults, memory leaks, and uses correct syntax. Function calls should not have data types; function headers and prototypes should match. The use of -> versus . should be clearly understood.

CS163: Data Structures

Students entering CS163 should already be fluent programming in C++ using classes, pointers and dynamic memory (new and delete), functions (using returned values, passing by value and passing by reference), and linear linked lists (traversal, creation, and removal); they should understand recursion at a conceptual level. Students should be proficient at programming pointer based linear linked list algorithms (traversal, insertion, and removal algorithms).

The goals of CS163 are to teach data structure algorithms; the course shows students how to efficiently apply recursion and select the appropriate data structure for a task. During the course, students learn how to implement solutions for circular linked list, doubly linked lists, arrays of linked lists, linked lists of arrays, and pointer based binary search trees. Students also learn to implement various abstractions including ordered lists, stacks, queues, hash tables, and other table abstractions including binary search trees, balanced trees and graphs. By the end of CS163, students should be proficient at applying recursion to data structure algorithms. This includes being fluent using functions, including the returned value of the function and returning values when appropriate.

After CS163, students should be able to write complete programs in C++, using multiple files (.cpp and .h files) implementing pointer based data structure solutions. Students should be competent using the linux environment developing software using **vi, vim, or emacs** without the assistance of an IDE, the internet, or external library support (beyond iostream, cctype, string and/or cstring libraries). Students should be fluent with the use of arguments (both pass by value and pass by reference) rather than relying on global variables. Good use of structured programming techniques is highly recommend to avoid common programming pitfalls.

Students taking PSU's CS163 course implement 5 individual programming assignments, which include developing the following from scratch: linear and doubly linked lists (ordered list), circular linked lists (queue), linear linked list of arrays (stack, ordered list), arrays of linked lists (hash table, adjacency list) and pointer based binary search trees. Students taking PSU's CS163 course also experience 10 labs during the term that provide hands on practice with multiple classes, the aforementioned data structures and recursion. This includes implementing recursive balanced tree algorithms. All work is done using linux; no IDEs are allowed.

PSU recommends the use of proficiency demos to determine the overall skill level of the students taking CS163 and recommends only passing students who show fluency with **recursive pointer based data structure** solutions. Students passing CS163 should be fluent using recursive solutions for linked data structures (linear, circular and doubly linked) by the midterm and fluent using recursive solutions for pointer based binary search trees by the final. Students should be fluent at problems such as (for example):

- Remove every leaf from a BST
- Return the largest item in a BST
- Calculate the height of a BST
- Finding how many times a matching item appears in a BST

When examining fluency, students should be able to develop code that is free from segmentation faults, memory leaks, and properly use recursion in the solution. The use of debuggers, such as gdb, is highly encouraged. Correct syntax is expected.

CS202: Programming Systems

Students entering CS202 should already be fluent programming in C++ using classes, pointers and dynamic memory (new and delete), functions (using returned values, passing by value and passing by reference), and recursive solutions to pointer based linked data structures. Students should be able to demonstrate proficiency at applying recursive solutions to insert, traverse or remove from pointer based linked data structures of: linear linked lists, circular linked lists, doubly linked lists, arrays of linked lists, and binary search trees.

The goals of CS202 are prepare students for PSU's upper division courses, understanding the "Programming Systems" expected by the Professors of the upper division courses. Students implement OOP designs using concepts such as multiple classes, inheritance, dynamic binding, copy constructors, initialization lists, and operator overloading. Students apply advanced data structures at least five times during the term which includes combinations of data structures learned in CS163 (Data Structures), such as lists of trees, trees of trees, arrays of lists, balanced tree and graph implementations.

CS202 also prepares students for the process of learning new programming languages that may be expected by the Professors of the upper division courses. Being the last class in which syntax may be taught (as an entire course), instruction and programming time is spent moving from C++ to other programming languages, such as Java. CS202 however is not a Java course. Students learn how to learn new languages and apply the concepts of inheritance, dynamic binding, and advanced data structures to other languages such as Java.

PSU expects that students are able to write complete programs in C++, using multiple files (.cpp and .h files) implementing pointer based data structure solutions, without the use of external libraries to assist with the data structures. Students should be competent using the linux environment without the assistance of an IDE, the internet, or external library support (beyond iostream, string and/or cstring libraries

Students taking PSU's CS202 course implement 5 individual programming assignments, which include developing object oriented solutions with advanced data structures from scratch. Students taking PSU's CS202 course also experience 10 labs during the term that provide hands on practice with inheritance, initialization lists, copy constructors, dynamic binding, exception handling, operator overloading, applying recursion in languages that do not support pass by reference, and implementing data structures in Java. All work is done using linux; no IDEs for C++ development.

PSU recommends the use of proficiency demos to determine the overall skill level of the students taking CS202 and recommends passing only students who show fluency with C++ pointer based data structure solutions. At midterm time these include recursive solutions for linear linked lists, circular linked lists, doubly linked lists, and arrays of linked lists. At final time these include pointer based data structure solutions for binary search trees; students are expected by this time to be fluent with linux editor functionality (search, replace, navigation) and gdb, implementing recursive solutions for problems similar to these (for example):

- Linear Linked list: Determine if there are duplicates
- Circular Linked list: Remove every occurrence of a particular item
- Doubly Linked list: Swap every other node
- Array of linked lists: Remove the last item in each linear linked list
- BST: Return the second largest item in a BST