Portland State University

Maseeh College of Engineering and Computer Science

Proficiency Examination Process

2016-2017

The Testing Schedule at CCUT

Prior to Arrival – CCUT will have students practice using linux without an IDE. Students should use vi, pico, nano, or emacs. Students should practice booting from the PSU Live USB drive which has the supplied data structures object code.

March 29 – PSU Faculty will visit CCUT

- PSU Faculty will CCUT students applying for upper division entry
- A practice session will take place; student should bring laptop computers
- PSU and CCUT Faculty will establish the schedule for Proficiency Examinations.

March 30 – Applicants will practice to prepare for the test.

- PSU Faculty will meet with first year CCUT students

March 31 – The proficiency examinations take place.

- Ten to Twelve students begin each hour, per a predetermined schedule.
- Proficiency Exams will not exceed 2 hours in length.
- PSU Faculty will deliver a report to CCUT and PSU on the results

PSU Expectations of Student Competencies

Students applying to PSU's Computer Science department must pass a proficiency examination. This examination has students demonstrate the ability to program, implement data structure algorithms and use recursion. Applicants will program independently while being observed by PSU Faculty, without any assistance.

PSU requires that applicants program in C++ using GNU's C++ compiler using C++ standard 98 on the linux command line, without the assistance of IDEs or other auxiliary (or online) materials. PSU expects students to show proficiency implementing recursive solutions to data structure problems.

The Process

The process of taking the proficiency exam starts with a practice session. During the practice session, applicants will experiment with the environment used for the proficiency exam and work through a sample problem.

The actual proficiency exam will take place for all applicants two days later. Students will receive a randomly assigned question and demonstrate programming in C++ with no additional external support. **Students must receive a passing score for their applications to be further processed by PSU.**

Computer Systems

Students will be booting laptop computers using a Linux LIVE bootable USB provided by PSU. Students will use **terminal** and a linux editor of vi, nano, pico or emacs.

1. It is expected that each CCUT applicant will have access to a laptop computer.
2. During the proficiency examination, CCUT applicants will use a Linux Live bootable USB drive provided by PSU Proficiency Testing Faculty.

3. **Compiler Requirements:**

   During the proficiency exam, students must use the GNU GCC C++ compiler (g++) in the default –ansi mode. This mode meets the **C++ standard 98** guidelines (which can be obtained through these flags: –std=c++98, -std=gnu++98, or -ansi).

   Students should compile with the –Wall flag to enable warnings about constructions that should be avoided. Students may compile with the –g flag to support the use of gdb. Use of any other flags is not allowed during the exam.

Expected Core Competencies

PSU expects that students applying for entrance into the upper division be 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 should be fluent with the use of arguments (both pass by value and pass by reference) without using global variables.

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. Correct syntax is expected.

PSU expects students to show fluency with C++ pointer based data structure solutions. This includes recursive solutions for linear linked lists, circular linked lists, doubly linked lists, and arrays of linked lists and 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 duplicate data within an existing linked lists.
- 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
- Binary Search Tree: Return the largest item in a binary search tree

Proficiency Exams – The Process:

1. Show Picture ID
2. It is expected that proficiency exams will be completed using the Roman Alphabet
    a. All **comments** must be in **English**
3. During the proficiency examination, no external programming materials may be used
    a. Dictionaries **may be** used
    b. Students **may not** access programming books, notes, files or programs
    c. No other electronic devices such as phones, smart watches may be used
    d. Online access to materials is prohibited
4. The proficiency demo question will be randomly assigned
5. Once a question has been selected, the student will be seated in the testing arena
    a. Follow the instructions given to boot the laptop and changed into the correct directory
    b. Students may use vi, nano, pico or emacs
    c. **We recommend** designing before coding.
    d. Students are being scored on their comprehensive approach to problem solving and implementing pointer based data structure algorithms.
    e. **Comment out code**. Never delete any code in the examination
    f. More than one function may be written
    g. Please **compile and run the code multiple times**
    h. Please **debug** using **gdb**

6. **Proficiency Exam - Rules:**
    - All problems will require a **recursive** solution
    - **NO LOOPS may be used in the solution,** unless indicated in the problem statement
    - **Global variables** are not allowed
    - Each problem will be given a public and private member function prototype
    - The public prototype supplied represents the function that should be called by main
    - The prototypes supplied may not be changed
    - For questions that have students count, average, or traverse in some way, the **solution should NOT modify the data structure**
    - For questions that ask students to insert or copy, the solution should **NOT delete items**
    - **As changes are made to the code,** please comment out code rather than deleting it!
    - Use of the cstring, cctype, and iostream libraries are allowed. No other libraries may be included or used.

Proficiency Exam – Sample Practice Questions:

A complete environment will be provided by PSU that already creates and populates the data structure with data. The data may be integer data or dynamically allocated arrays of characters. Students will be implementing specific functions to perform a task which will then be linked with code to build, display, and destroy the data structure assigned.

Linear linked lists, circular and doubly linked lists:

1. Write a recursive function in C++ to find duplicate data within an existing linked list.
2. Write a recursive function in C++ to remove all nodes in that linked list
3. Write a recursive function in C++ to remove all nodes except the last two nodes
4. Write a recursive function in C++ to add a node to the end but only if it doesn't already exist in the linked list
5. Write a recursive function in C++ to move the last node to the beginning of the linked list
6. Write a recursive function in C++ to make a complete copy of a linked list
7. Write a recursive function in C++ to copy a circular linked list into a linear linked list

Binary search trees:

1. Write a recursive function in C++ to make a copy of a binary search tree
2. Write a recursive function in C++ to make a copy of a binary search tree and place it in a linear linked list, sorted
3. Write a recursive function in C++ to add a node into a binary search tree
4. Write a recursive function in C++ to remove the largest item in a binary search tree
5. Write a recursive function in C++ to remove the largest two items in a binary search tree
6. Write a recursive function in C++ to find the root’s in-order successor