Assessment of CPSC 131 during Spring 2019

Anand Panangadan
September 13, 2019
CPSC 131 “Data Structures”

• Part of the introductory 120-121-131-301/EPP sequence
  • Required of all CPSC majors, CPSC minors, CpEng majors
  • Somewhat popular among business, math students

• CPSC 131 is “team-taught”
  • 6 sections
  • 5 instructors
    • Doina Bein, Thomas Bettens, Allen Holliday, David McLaren, Anand Panangadan
  • ~180 students
  • All sections share syllabus, homework, projects, and exams
    • minor variations for exams
## Course mapping table

<table>
<thead>
<tr>
<th>Core Course CPSC</th>
<th>SLO 1</th>
<th>SLO 2</th>
<th>SLO 3</th>
<th>SLO 4</th>
<th>SLO 5</th>
<th>SLO 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>120</td>
<td>CODE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>121</td>
<td>ACODE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>131</td>
<td>ALG</td>
<td>ACODE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>223</td>
<td>CODE,TEST</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>240</td>
<td>HW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>311</td>
<td>WRITE, SPEAK</td>
<td>FDBK</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>315</td>
<td>WRITE, SPEAK</td>
<td>ETH, IPSEC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>323</td>
<td>DESC</td>
<td></td>
<td></td>
<td></td>
<td>COOP, FDBK</td>
<td></td>
</tr>
<tr>
<td>332</td>
<td>RESPEC</td>
<td></td>
<td></td>
<td></td>
<td>COOP, FDBK</td>
<td></td>
</tr>
<tr>
<td>335</td>
<td>DESC</td>
<td></td>
<td></td>
<td></td>
<td>ALG</td>
<td></td>
</tr>
<tr>
<td>351</td>
<td>HW</td>
<td>DESC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>362</td>
<td>RESPEC</td>
<td>DESC, TEST</td>
<td></td>
<td>PROC</td>
<td>RESPEC, DESC, Test</td>
<td></td>
</tr>
<tr>
<td>440</td>
<td>HW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>471</td>
<td>HW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>481</td>
<td>RESPEC</td>
<td>DESC, TEST</td>
<td></td>
<td>ETH, IPSEC</td>
<td>PROC</td>
<td>RESPEC, DESC, TEST, FB</td>
</tr>
</tbody>
</table>

Exit Survey

| | | | | ETH, IPSEC | PROC | RESPEC, DESC, TEST, FB |
Performance Indicators

• **ACODE**: Write syntactically-correct and more advanced, nuanced C++ programming source code that make appropriate use of object-oriented concepts such as classes, encapsulation, and templates; and includes pointers, recursion, and memory management.

• **ALG**: Design an algorithm to solve a novel computational problem that builds upon classical techniques (e.g. data structures, discrete mathematics tools, divide-and-conquer, dynamic programming) and analyze the algorithm in terms of formalisms such as asymptotic efficiency, lower bounds, or computational complexity.
Before the course begins: in the syllabus

Learning Goals:
• Analyze an algorithm or procedure and derive its time efficiency class in terms of asymptotic notation.
• Design and/or implement software that makes effective and appropriate use of fundamental data structures (e.g. stack, queue, search tree, hash table).
• Identify possible solutions to a problem and analyze their feasibility or trade-offs.
• Write syntactically-correct source code, making appropriate use of fundamental constructs such as variables, branches, loops, and functions that solves a well-posed computational problem.
Structure of the course

- Homework (weekly)
- Programming projects (every 2 weeks)
- Reading assignments (online textbook)
- Midterm (on paper)
- Final (on paper)
- Class participation

Assessed ACODE in 4th project
Assessed ALG in final
Deciding Satisfactory, Developing, Unsatisfactory boundaries

• We decided to set simple point boundaries for each question/project to be used for assessment

• Example: for ACODE
  • 4th project maximum score: 100
  • 80-100: Satisfactory
  • 50-79: Developing
  • 0-49: Unsatisfactory
  • No attempt: did not include in data
Performance Indicator (PI): ACODE

- Project distributed using Github Classroom
- Completed code downloaded, compiled, and run automatically
- 80/100 points were given based on automatic test
  - 20/100: instructor’s checks – comments, memory leaks

Project 4: Gate Control
This project implements software to control access to a company’s facility. The company has several buildings and a parking lot for its staff, all surrounded by a fence that has an electrically-operated gate for employees to drive through if they have an authorized access card. The gate has a card reader that is connected to a computer that will receive a card’s identifying number, validate the access, and operate the gate operator’s motor to open the gate. The computer will record each transaction—the date and time of each access, allowed or denied.

...
Performance Indicator (PI) Rubric: ACODE

• UNSATISFACTORY: Unable to develop C++ programming source code that make appropriate use of object-oriented concepts and follow acceptable coding standards or conventions
  • < 50

• DEVELOPING: Able to write C++ programming source code that make use of some object-oriented concepts and follow some coding conventions
  • 50-79

• SATISFACTORY: Able to demonstrate the ability to write C++ programming source code that make appropriate use of object-oriented concepts such as classes, encapsulation, and templates; includes pointers, recursion, and memory management; and follow industry coding standards or conventions
  • 80-100
Testing AddAuthorization
Table initially empty (5 points) Pass. 5/80 points
Add first, success/failure status (1 point) Pass. 6/80 points
Add first, table check (5 points) Pass. 11/80 points
Add second, success/failure (1 point) Pass. 12/80 points
Add second, table check (4 points) Pass. 16/80 points
Add duplicate (6 points) Pass. 22/80 points
Testing DeleteAuthorization
Existing Card success/fail status (1 point) Pass. 23/80 points
Existing card, table check (5 points) Pass. 28/80 points
Non-existent card (6 points) Pass. 34/80 points
Testing ChangeAuthorization
Existing card, success/fail status (1 point) Pass. 35/80 points
Existing card, table check (5 points) Pass. 40/80 points
Non-existent Card (6 points) Pass. 46/80 points
Testing Access
Allowed 1 (6 points) Pass. 52/80 points
Allowed 2 (5 points) Pass. 57/80 points
Denied 1 (6 points) Pass. 63/80 points
Denied 2 (5 points) Pass. 68/80 points
Testing Card-Specific Functions
GetCardAuthorization, success/failure status (1 point) Pass. 69/80 points
GetCardAuthorization, record check (5 point) Pass. 74/80 points
GetCardTransactions (6 points) Pass. 80/80 points

Final result: 80/80 points

Excellent work again! You have also developed an elegant programming style.
Total: 80+20 = 100

```cpp
bool GateControl::AccessAllowed(CardNumber number) {
    if (authorizationMap_.find(number) == authorizationMap_.end()) {
        // If there is no entry for that number, set name to *** and add the rest of info to log. Returning false, cause not found.
        Transaction temp(number, "***", gCurrentDate, gCurrentTime, false);
        transactionVector_.push_back(temp);
        return false;
    }
    // Other checks...
}
```
Performance Indicator (PI): ALG

• Question on the final exam
• Created specifically to assess this PI
• Points: 5/100

V. Problem solving (5 points total)
You are asked to develop a computer program to create a book index: the list of words that appear in the book in alphabetical order and the page numbers they appear on. The input to the program will be the book in text form: the sequence of words and the page numbers. Describe in 4-5 sentences the design of your program to create the book index. Include in your description, the important classes, data structures, and algorithms in this application.
Performance Indicator (PI) Rubric: ALG

• UNSATISFACTORY: Unable to design a process or algorithm
  • 0-2 points

• DEVELOPING: Able to design a semi-complete and/or semi-correct algorithm
  • 3 points

• SATISFACTORY: Able to design a correct algorithm to solve a novel computational problem
  • 4-5 points
Instructor feedback

• We scanned the relevant pages from the final exams of selected students
  • one sample for Unsatisfactory, Developing, Satisfactory
• Save as PDF
• upload as part of assessment samples
Student work as evidence in CPSC 131

- For evidence, student work needs to include instructor feedback
- **ACODE**
  - C++ code submitted via Github but entered feedback on Titanium
  - To prepare a PDF file for digital evidence:
    - We copied C++ code into MS Word, added feedback at the top and converted to PDF
    - ONLY for samples (one each for Satisfactory, Developing, Unsatisfactory)
- **ALG**
  - Scanned relevant pages from the final exams
Numerical data

ACODE
Total number of students: 20
Satisfactory 6
Developing 3
Unsatisfactory 11

ALG
Total number of students: 23
Satisfactory 13
Developing 3
Unsatisfactory 7
Entering assessment data online