PROG20799
Data Structures & Algorithm Development - C

I: Administrative Information **II: Course Details** **III: Topical Outline(s)** **Open in Word**

Retain during the course and for future use when applying for credit at other educational institutions

Land Acknowledgement

Sheridan College resides on land that has been, and still is, the traditional territory of several Indigenous nations, including the Anishinaabe, the Haudenosaunee Confederacy, the Wendat, and the Mississaugas of the Credit First Nation. We recognize this territory is covered by the Dish with One Spoon treaty and the Two Row Wampum treaty, which emphasize the importance of joint stewardship, peace, and respectful relationships.

As an institution of higher learning Sheridan embraces the critical role that education must play in facilitating real transformational change. We continue our collective efforts to recognize Canada's colonial history and to take steps to meaningful Truth and Reconciliation.

Section I: Administrative Information

Program(s): CST-Information Systems Eng, CST-Software Dev & Network Eng, CST-Software Engineering, Computer Programming
Program Coordinator(s): Simon Hood, Ann Cadger, Walid Belal, Satyendra Narayan
Course Leader or Contact: Haya El Ghalayini
Version: 20240108_01
Status: Approved (APPR)

Section I Notes: N/A

Total hours: 84.0
Credit Value: 6.0
Credit Value Notes: N/A
Effective: Winter 2024
Prerequisites: (PROG32758) OR (PROG12583)
Corequisites: N/A
Equivalents: N/A
Pre/Co/Equiv Notes: N/A

Section II: Course Details

Detailed Description

Students develop skills and knowledge of essential computer science data structures and algorithms through the C programming language. They are introduced to various data structure concepts: stacks, queues, linked lists, recursion, trees, hashes, and graphs. In addition, students explore more advanced topics including sorting and searching algorithms, algorithm complexity analysis, and Big O notation. They gain hands-on experience with concepts, techniques, and strategies necessary to write, compile, and execute efficient programs, moving beyond simple programming solutions into more complex solutions that also consider programming time complexity and the most effective storage structures.

Program Context

CST-Information Systems Eng **Program Coordinator(s):** Walid Belal
This required course introduces students to more complex computer programming subjects. Students explore fundamental algorithms, including standard search and sort algorithms, as well as storage and programming concepts beyond simple arrays like lists, stacks, queues, trees, and graphs. The C programming language is introduced to facilitate additional learning of memory management and debugging, and students are encouraged not just to solve problems, but to solve them using more efficient techniques, more appropriate structures, and more

efficient algorithms than they have in previous studies.

CST-Software Dev & Network Eng **Program Coordinator(s):** Simon Hood

This required course introduces students to more complex computer programming subjects. Students explore fundamental algorithms, including standard search and sort algorithms, as well as storage and programming concepts beyond simple arrays like lists, stacks, queues, trees, and graphs. The C programming language is introduced to facilitate additional learning of memory management and debugging, and students are encouraged not just to solve problems, but to solve them using more efficient techniques, more appropriate structures, and more efficient algorithms than they have in previous studies.

CST-Software Engineering **Program Coordinator(s):** Satyendra Narayan

This required course introduces students to more complex computer programming subjects. Students explore fundamental algorithms, including standard search and sort algorithms, as well as storage and programming concepts beyond simple arrays like lists, stacks, queues, trees, and graphs. The C programming language is introduced to facilitate additional learning of memory management and debugging, and students are encouraged not just to solve problems, but to solve them using more efficient techniques, more appropriate structures, and more efficient algorithms than they have in previous studies.

Computer Programming **Program Coordinator(s):** Ann Cadger, Satyendra Narayan

This required course introduces students to more complex computer programming subjects. Students explore fundamental algorithms, including standard search and sort algorithms, as well as storage and programming concepts beyond simple arrays like lists, stacks, queues, trees, and graphs. The C programming language is introduced to facilitate additional learning of memory management and debugging, and students are encouraged not just to solve problems, but to solve them using more efficient techniques, more appropriate structures, and more efficient algorithms than they have in previous studies.

Course Critical Performance and Learning Outcomes

Critical Performance:

By the end of this course, students will have demonstrated the ability to develop efficient algorithms utilizing a variety of data structures using the C programming language.

Learning Outcomes:

To achieve the critical performance, students will have demonstrated the ability to:

1. Apply C dynamic memory management techniques to create and destroy data structures.
2. Describe the Big O notation for the analysis of algorithms.
3. Determine the efficiency of algorithms based on code presentation.
4. Use C to implement data structures and abstract data types.
5. Identify data structures for solving specific problems based on problem requirements.
6. Analyze complexity of sorting and searching algorithms.
7. Implement sorting and searching algorithms in C.

Evaluation Plan

Students demonstrate their learning in the following ways:

Evaluation Plan: IN-CLASS

Assignments (4 @ 7.5%)	30.0%
Quizzes (4 @ 2.5%)	10.0%
Tests (4 @ 15%)	60.0%
Total	100.0%

Evaluation Notes and Academic Missed Work Procedure:
To pass the course, students must achieve a 50% weighted average across the tests and the exams and at least 50% overall in the course.

Students must submit/complete all assignments, in-class activities and projects by the scheduled due date and write all tests on the specified date/time. Exceptions will only be made under extraordinary circumstances.

Refer to the School of Applied Computing's Academic Procedures for Evaluations for more details regarding missed work: Procedures for Evaluations

Provincial Context
The course meets the following Ministry of Colleges and Universities requirements:

Essential Employability Skills
Essential Employability Skills emphasized in the course:

- Communication Skills - Communicate clearly, concisely and correctly in the written, spoken, visual form that fulfills the purpose and meets the needs of the audience.
- Communication Skills - Respond to written, spoken, or visual messages in a manner that ensures effective communication.
- Critical Thinking & Problem Solving Skills - Use a variety of thinking skills to anticipate and solve problems.
- Critical Thinking & Problem Solving - Apply a systematic approach to solve problems.
- Information Management Skills - Analyze, evaluate, and apply relevant information from a variety of sources.
- Interpersonal Skills - Show respect for the diverse opinions, values, belief systems, and contributions of others.
- Information Management - Locate, select, organize and document information using appropriate technology and information systems.
- Interpersonal Skills - Interact with others in groups or teams in ways that contribute to effective working relationships and the achievement of goals.
- Numeracy - Execute mathematical operations accurately.
- Personal Skills - Take responsibility for one's own actions, decisions, and consequences.
- Personal Skills - Manage the use of time and other resources to complete projects.

Prior Learning Assessment and Recognition
PLAR Contact (if course is PLAR-eligible) - Office of the Registrar
Students may apply to receive credit by demonstrating achievement of the course learning outcomes through previous relevant work/life experience, service, self-study and training on the job. This course is eligible for challenge through the following method(s):

- Challenge Exam
  Notes: Both an interview and a challenge exam must be completed.
- Interview
  Notes: Both an interview and a challenge exam must be completed.

Section III: Topical Outline

Some details of this outline may change as a result of circumstances such as weather cancellations, College and student activities, and class timetabling.

Instruction Mode: In-Class
Professor: Multiple Professors
Resource(s):
Course material costs can be found through the Sheridan Bookstore

	Type	Description
Optional	Textbook	Advanced Topics in C: Core Concepts in Data Structures., Kalicharan, N., Apress, 2013, Available through the Sheridan Library resource Books 24x7 service.

Applicable student group(s): Computer Systems Technician - Software Engineering, Computer Systems Technology - Software Development and Network Engineering, Computer Programmer, Computer Systems Technology - Information Systems Engineering
Course Details:

Module 1. Introduction to C, data structures, and the course
- Fundamentals of data structures
- C fundamentals and compiler setup
- Arrays and strings
- User-defined types
(Assignment 1 @ 7.5%)
(Quiz 1 @ 2.5%)

Module 2. Pointers and dynamic memory management
- Introduction to pointers
- More pointers
- Pointers and dynamic memory management
- Dynamic arrays
(Test 1 @ 15%)

Module 3. Linked lists
- Operations
- Implementations
- Applications
(Assignment 2 @ 7.5%)
(Quiz 2 @ 2.5%)

Module 4. Stacks and queues
- Array-based implementations
- List-based implementations
(Test 2 @ 15%)

Module 5. Searching and sorting
- Complexity analysis
- The Big O notation
- Searching and data structures
- Sorting algorithms
(Quiz 3 @ 2.5%)
(Assignment 3 @ 7.5%)
(Test 3 @ 15%)

Module 6. Trees
- Binary trees
- Searching and sorting
- Balancing
(Assignment 4 @ 7.5%)

Module 7. Hashtables
- Implementation
- Complexity analysis
(Quiz 4 @ 2.5%)

Module 8. Graphs
- Implementation
- Searching algorithms
(Test 4 @ 15%)

Sheridan Policies

It is recommended that students read the following policies in relation to course outlines:

- **Academic Integrity**
- **Copyright**
- **Intellectual Property**
- **Respectful Behaviour**
- **Accessible Learning**

All Sheridan policies can be viewed on the Sheridan policy website.

Appropriate use of generative Artificial Intelligence tools: In alignment with Sheridan's Academic Integrity Policy, students should consult with their professors and/or refer to evaluation instructions regarding the appropriate use, or prohibition, of generative Artificial Intelligence (AI) tools for coursework. Turnitin AI detection software may be used by faculty members to screen assignment submissions or exams for unauthorized use of artificial intelligence.

Course Outline Changes: The information contained in this Course Outline including but not limited to faculty and program information and course description is subject to change without notice. Nothing in this Course Outline should be viewed as a representation, offer and/or warranty. Students are responsible for reading the Important Notice and Disclaimer which applies to Programs and Courses.