UNIVERSITY OF SASKATCHEWAN
Department of Computer Science
176 Thorvaldson Building
110 Science Place Saskatoon, SK S7N 5C9 Canada
Telephone: (306) 966-4886 Facsimile: (306) 966-4884

COURSE SYLLABUS
CMPT 145.3: PRINCIPLES OF COMPUTER SCIENCE

***Note**: This is a preliminary version of the syllabus. Some details relating to the course offering (e.g., the number of assignments, or the weight of the final exam) may change slightly with the final version. This information is provided as a convenience for students, and should not be taken as the final word as long as this notice appears.*

***Note:** This course replaces CMPT 115 in several degree programs, but is not an exact equivalent.*

Catalogue Description:

This course builds on CMPT 141 by introducing additional problem solving methods and computer science principles, to solve larger problems that are more data intensive, or require more sophisticated techniques. These principles include data structures for efficient storage and retrieval of data, selection of appropriate data structures, algorithmic paradigms for solving difficult problems, and analysis of algorithms' time and space requirements. This course also emphasizes fundamental principles of coding style, testing, and top-down design for writing robust, maintainable software.

Prerequisite(s):	CMPT 141; or CMPT 111 and permission of the department. **Note**: Students with credit for CMPT 270 cannot take CMPT 145 for credit.
Class Time & Location:	TBA
Website:	TBA

Instructor Information

Instructor:	Michael C Horsch
Contact:	Email: `horsch@cs.usask.ca` Office Phone: 966-2161
Office Hours:	Location: TBA Hours: TBA

Course Objectives

By the end of this course, you are expected to be able to:

- Be proficient in writing robust, maintainable software in Python.
- Design algorithms using pseudo-code, and analyze algorithms written in pseudo-code.
- Analyze time and space complexity of algorithms, and to compare and evaluate algorithms and data structures in terms of complexity analysis.
- Explain the concept of abstract data types (ATDs) in terms of interface and encapsulation.
- Describe the use and behavior of objects in Python, as examples of the ADT concept.

- Apply specific data types: arrays, trees, binary search trees as part of the solution to computational problems.
- Apply recursion to computational tasks involving data structures such as lists and trees.
- Apply skills in elementary software design, including test-driven design, and iterative development.
- Apply skills in elementary software testing, including black-box testing, and boundary conditions, and debugging.
- Describe and apply strategies such as divide and conquer, greedy algorithms, and backtracking.

Course Overview

This course can be taken as a Science credit for Arts & Science majors, and is also a required course in Computer Science major programs, and a few other programs. Lectures will be opportunities to apply the concepts covered in the course, discuss them, as well as to ask questions and receive guidance; we will not waste class time reading PowerPoint slides to you. Short readings will be assigned before each class, and you will be expected to be prepared to discuss, ask questions, and participate. Laboratories are your opportunities to put the week's material into practice under the guidance of a teaching assistant.

- Assignments are weekly, to ensure that all the relevant material is put into regular, consistent practice. Even a simple assignment can turn into a time-consuming affair, if you get stuck on something that blocks your progress. Working at the last minute is a guaranteed source of stress and burn-out. To manage your workload you must practice effective time management. Start every assignment early, to allow yourself time to consult if you run into a problem. Please make use of the teaching resources (instructors' office hours, TAs, labs, lectures, discussion forums etc.) available to you.

Students who complete CMPT 145 with diligence will be able to:

- Build substantial applications making use of Python's extensive libraries.
- Build computational solutions to a wide variety of problems, using a range of algorithmic strategies, and a range of data structures.
- Verify that Python programs work correctly.
- Assess and mitigate computational efficiency concerns that may arise in practice.
- Continue their formal study of computer science in courses such as CMPT 214, CMPT 260, and CMPT 270.

ACM-2013 Learning Outcomes

This course achieves the following learning outcomes (listed alphabetically) from the ACM-2013 Computer Science Curriculum Guidelines (https://www.acm.org/education/CS2013-final-report.pdf):

- Implement, test, and debug simple recursive functions and procedures. [Usage, SDF-AD-5]
- Determine whether a recursive or iterative solution is most appropriate for a problem. [Assessment, SDF-AD-6]
- Identify the data components and behaviors of multiple abstract data types. [Usage, SDF-AD-9]
- Implement a coherent abstract data type, with loose coupling between components and behaviors. [Usage, SDF-AD-10]
- Design, implement, test, and debug a program that uses each of the following fundamental programming constructs: basic computation, simple I/O, standard conditional and iterative structures, the definition of functions, and parameter passing. [Usage, SDF-FPC-5]
- Discuss the appropriate use of built-in data structures. [Familiarity, SDF_FDS-1]
- Describe common applications for each of the following data structures: stack, queue, priority queue, set, and map. [Familiarity, SDF_FDS-2]

- Write programs that use each of the following data structures: arrays, records/structs, strings, linked lists, stacks, queues, sets, and maps. [Usage, SDF_FDS-3]
- Compare alternative implementations of data structures with respect to performance. [Assessment, SDF_FDS-4]
- Choose the appropriate data structure for modeling a given problem. [Assessment, SDF_FDS-7]
- Explain why the creation of correct program components is important in the production of high-quality software. [Familiarity, SDF-DM-2]
- Apply a variety of strategies to the testing and debugging of simple programs. [Usage, SDF-DM-8]
- Construct, execute and debug programs using a modern IDE and associated tools such as unit testing tools and visual debuggers. [Usage, SDF-DM-9]
- Construct and debug programs using the standard libraries available with a chosen programming language. [Usage, SDF-DM-10]
- Explain what is meant by "best" and "worst" case behavior of an algorithm. [Partial, AL-BA-1]
- Determine informally the time and space complexity of simple algorithms. [Usage, AL-BA-3]
- List and contrast standard complexity classes. [Familiarity, AL-BA-5]
- Perform empirical studies to validate hypotheses about runtime stemming from mathematical analysis. Run algorithms on input of various sizes and compare performance. [Assessment, AL-BA-6]
- For each of the strategies (brute-force, greedy, divide-and-conquer, recursive backtracking, and dynamic programming), identify a practical example to which it would apply. [Familiarity, AL-AS-1]
- Use a greedy approach to solve an appropriate problem and determine if the greedy rule chosen leads to an optimal solution. [Assessment, AL-AS-2]
- Use a divide-and-conquer algorithm to solve an appropriate problem. [Usage, AL-AS-3]
- Use recursive backtracking to solve a problem such as navigating a maze. [Usage, AL-AS-4]
- Implement simple search algorithms and explain the differences in their time complexities. [Assessment, AL-FDSA-2]
- Explain how tree balance affects the efficiency of various binary search tree operations. [Familiarity, AL-FDSA-7]

Student Evaluation

Note: All students must be properly registered in order to attend lectures and receive credit for this course.

Grading Scheme (preliminary)

8 Assignments (4% each)	32%
Tutorial Exercises (1% each)	8%
Midterm Exam	25%
Final Exam	35%
Total	**100%**

- There will be 8 **assignments** in this course, one approximately every week. Assignments will consist of two portions:
  1. written questions that require written answers;
  2. programming questions which require you to write computer programs.

  Submission instructions will be included with each assignment description. Generally, you will upload your solutions as files to Moodle, unless you are instructed otherwise. Generally, text files are preferred to documents that include formatting (e.g., MSWord documents). A document that cannot be opened will

receive a grade of zero; do not assume the markers will take the time to open your file if it is in a file format that is not standard.

- **Tutorials** have associated exercises, which we expect you to do for the skills you learn by doing them. You are expected to submit your tutorial exercises weekly, along with your assignment solutions. These exercises will be graded, and will make up 8% of your total course grade. You should expect to complete these exercises in the time allotted for tutorials, and should not require any time outside the tutorials.
- The **midterm exam** will be held in mid February.

  The midterm examination is written, closed-book; only bring water, your student card, and writing instruments. The midterm examination will have some multiple choice, and some programming, and short answer questions. The mid-term examination is intended to provide practice for the final exam, and to provide feedback to students regarding their current performance.

  Please see the section on Policies for important information concerning missed midterms and final exams. In the case of a missed midterm, the instructor, in consultation with the student, will determine how the missed work will be compensated for; one potential alternative is transferring the weight of the midterm onto the final examination.
- The final examination will be scheduled by central timetabling to occur during the usual final examination interval. It will be three hours long, written, closed-book; bring only water, your student card, and writing instruments.

Please see the section on Policies for important information concerning missed midterms and final exams.

Criteria that must be met to Pass

Students must write the final exam. A student who does not write the final exam will receive a grade of at most 49 in the course.

Attendance Expectation:

- Attend every class, and participate actively. There will be short reading assignments for all classes (see Moodle webpage http://moodle.cs.usask.ca), and students are expected to come to class having completed the readings. There is no penalty for missed lectures.
- Attend all laboratory sessions. These are opportunities to practice the course material with the guidance of a teaching assistant. There is no penalty for missed lab sessions, provided that the lab exercises are completed by the due date.
- Attend the midterm examination. If you have part-time work, or other responsibilities, please try to make arrangements ASAP that will allow you to write the midterm. We will make alternative arrangements for students who cannot attend the evening seatings, but obviously, we would like to keep the special arrangements to a minimum. A missed midterm will be counted as a score of zero, excluding exemptions due to health or compassionate grounds.

Final Exam Scheduling:

The Registrar schedules all final examinations, including deferred and supplemental examinations. Students are advised not to make travel arrangements for the exam period until the official exam schedule has been posted.

Tutorials and Help Sessions

Tutorials

Tutorials in a laboratory setting are mandatory and include new material not presented in class. Lectures

emphasize the data organization concepts using pseudocode; tutorials focus on how to implement, in C++, the concepts studied in lecture. Material presented in tutorials may appear on examinations. If you miss a tutorial section, you may try to attend another section during that week; but there is considerable risk you will not be able to find a seat.

Help Sessions

Obstacles to progress and completion of assignments can sometimes be part of the homework (i.e., something we want you to think about carefully), and sometimes beyond the scope of the course (i.e., a problem that you can't really be expected to manage in first year), and it can be nearly impossible to tell the difference without some advice from a TA or instructor.

There are several help sessions in the Spinks Computer Lab that are specifically for CMPT 145 students. The TAs are all prepared for the assignments and lab questions. We highly recommend you to work in the computer lab, because it is very helpful if you can get help when you have difficulties. The schedule will be announced in the first two weeks of the term on Moodle.

Textbook Information

Extensive notes will be provided on the course Moodle site.

Required Texts

- **TBA.**

Recommended Text

- TBA

Useful Texts

- Goodrich, Tamassia, and Goldwater. Data Structures and Algorithms in Python. Wiley, 2013. **ISBN-10:** 1118290275
- Sedgewick, Wayne and DOndero. Introduction to Programming in Python. Addison-Wesly, 2015.

Lecture Schedule

- Week 1: Introduction to software design principles
- Week 2: Programming practice: Good style, defensive programming
- Week 3: Stacks and queues, and their applications.
- Week 4: Algorithm analysis, or why your code isn't as good as you thought!
- Week 5: Abstract data types for design and testing. Testing your code.
- Week 6: Trees, binary trees, and binary search trees, and their applications.
- Week 7: More software development techniques: iterative design, test-driven design
- Week 8: Objects!
- Week 9: Getting the most out of Python: APIs and interfaces.
- Week 10: Techniques: greedy algorithms, and back-tracking.
- Week 11: Looking beneath Python: understanding the machine leads to speed when you need it.
- Week 12: Looking beyond Python: other languages

Policies

Recording of Lectures

Lecture videos will be available through a link on the course Moodle site.

Late Assignments

Due to the weekly assignment schedule, late submissions cannot be accepted. Be sure to start your assignments early, and hand in partial solutions for partial credit if necessary. We understand that legitimate, exceptional circumstances sometimes prevent a deadline from being met. If you feel you cannot submit an assignment on time, please talk to the course instructor at least a day before the assignment is due. Extensions on assignments will be granted only by the course instructor. All extension requests will require suitable documentation.

Missed Assignments

Students are expected to attempt (and hopefully complete!) all assignments, and all laboratory exercises. It's better to submit partially completed assignments than to submit nothing at all. A missed assignment will receive a score of zero. If you miss an assignment for medical or compassionate reasons, contact your instructor as soon as possible.

Missed Examinations

1. Students who miss an exam should contact the instructor as soon as possible. If it is known in advance that an exam will be missed, the instructor should be contacted before the exam.

2. "A student who is absent from a final examination due to medical, compassionate, or other valid reasons, may apply to the College of Arts and Science Undergraduate Student's Office for a **deferred** exam. **Application must be made within three business days of the missed examination** and be accompanied by supporting documents." (http://artsandscience.usask.ca/students/help/success.php)

Incomplete Course Work and Final Grades

"*When a student has not completed the required course work, which includes any assignment or examination including the final examination, by the time of submission of the final grades, they may be granted an extension to permit completion of an assignment, or granted a deferred examination in the case of absence from a final examination.*

Extensions past the final examination date for the completion of assignments must be approved by the Department Head, or Dean in non-departmentalized Colleges, and may exceed thirty days only in unusual circumstances. The student must apply to the instructor for such an extension and furnish satisfactory reasons for the deficiency. Deferred final examinations are granted as per College policy.

In the interim, the instructor will submit a computed percentile grade for the class which factors in the incomplete coursework as a zero, along with a grade comment of INF (Incomplete Failure) if a failing grade.

In the case where the student has a passing percentile grade but the instructor has indicated in the course outline that failure to complete the required coursework will result in failure in the course, a final grade of 49% will be submitted along with a grade comment of INF (Incomplete Failure).

If an extension is granted and the required assignment is submitted within the allotted time, or if a deferred examination is granted and written in the case of absence from the final examination, the instructor will submit a revised assigned final percentage grade. The grade change will replace the previous grade and any grade comment of INF (Incomplete Failure) will be removed.

A student can pass a course on the basis of work completed in the course provided that any incomplete course work has not been deemed mandatory by the instructor in the course outline and/or by College regulations for achieving a passing grade." (http://policies.usask.ca/policies/academic-affairs/academic-courses.php)

For policies governing examinations and grading, students are referred to the Assessment of Students section of the University policy "Academic courses: class delivery, examinations, and assessment of student learning" (http://policies.usask.ca/policies/academic-affairs/academic-courses.php)

Academic Honesty

The University of Saskatchewan is committed to the highest standards of academic integrity and honesty. Students are expected to be familiar with these standards regarding academic honesty and to uphold the policies of the University in this respect. Students are particularly urged to familiarize themselves with the provisions of the Student Conduct & Appeals section of the University Secretary Website and avoid any behavior that could potentially result in suspicions of cheating, plagiarism, misrepresentation of facts and/or participation in an offence. Academic dishonesty is a serious offence and can result in suspension or expulsion from the University.

All students should read and be familiar with the Regulations on Academic Student Misconduct (http://www.usask.ca/secretariat/student-conduct-appeals/StudentAcademicMisconduct.pdf) as well as the Standard of Student Conduct in Non-Academic Matters and Procedures for Resolution of Complaints and Appeals (http://www.usask.ca/secretariat/student-conduct-appeals/StudentNon-AcademicMisconduct.pdf) Academic honesty is also defined and described in the Department of Computer Science statement on Academic Honesty (http://www.cs.usask.ca/students/academic-honesty/index.php).

For more information on what academic integrity means for students see the Student Conduct & Appeals section of the University Secretary Website at:
http://www.usask.ca/secretariat/student-conduct-appeals/forms/IntegrityDefined.pdf

Examinations with Disability Services for Students (DSS)

Students who have disabilities (learning, medical, physical, or mental health) are strongly encouraged to register with Disability Services for Students (DSS) if they have not already done so. Students who suspect they may have disabilities should contact DSS for advice and referrals. In order to access DSS programs and supports, students must follow DSS policy and procedures. For more information, check http://students.usask.ca/health/centres/disability-services-for-students.php, or contact DSS at 966-7273 or dss@usask.ca.

Students registered with DSS may request alternative arrangements for mid-term and final examinations. Students must arrange such accommodations through DSS by the stated deadlines. Instructors shall provide the examinations for students who are being accommodated by the deadlines established by DSS.