# http://www.ric.edu/webcommunications/images/SealWithText_Small_Black.pngUNDERGRADUATE CURRICULUM COMMITTEE (UCC) PROPOSAL FORM

## Cover page scroll over blue text to see further important [instructions](#instructions): [if not working select “COMMents on rollover” in your Word preferences under view] please read these.

**N.B. Please do not use highlight to select choices within a category but simply delete the options that do not apply to your proposal (e.g. in A.2 if this is a course revision proposal, just delete the creation and deletion options and the various program ones, so it reads “course revision”) Do not ever delete any of the numbered categories—if they do not apply leave them blank. ALL numbered categories in section (A) must be completed. If there are no resources impacted it is okay to put “none” in A. 7**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A.1. [Course or program](#Proposal) | **CSCI 209 DISCRETE STRUCTURES USING PYTHON** | | | |  |
| [Replacing](#Ifapplicable) |  | | | |
| A. 1b. Academic unit | **Faculty of Arts and Sciences** | | | |  |
| A.2. [Proposal type](#type) | **Course: creation** | | | |  |
| A.3. [Originator](#Originator) | **NAMITA SARAWAGI** | [Home department](#home_dept) | **COMPUTER SCIENCE AND INFORMATION SYSTEMS** | | |
| A.4. [Context and Rationale](#Rationale)  Note: Must include this additional information for all [new programs](#type) | This new course, titled Discrete Structures using Python, is being proposed as a required course in both the BA and BS Computer Science programs.  Discrete structures is a knowledge area for computer science as proposed in the ACM Curriculum Guidelines that are followed by Computer Science programs all over the world.  (Page 76: <https://www.acm.org/binaries/content/assets/education/cs2013_web_final.pdf>)  The objective of this course is to better prepare CS students early in major for their later CS courses: the computer programming sequence (CSC 211 and CSCI 212), as well as upper-level courses, by applying basic/foundational mathematical concepts and techniques to problems in, and applications of, computer science. The emphasis will be on computer science applications and problems. The foundational mathematics required to solve these problems and/or understand these applications will be taught. A basic course in high school or college algebra (Math 120) is sufficient math pre-requisite for this course, as mentioned in the ACM Curriculum Guidelines.  Students will require knowledge of basic computer programming (CSCI 157 – Introduction to Algorithmic Thinking in Python.) as some of the applications covered will come from programming language concepts. Other applications will come from other areas of computer science, such as cryptography, networks, compilers, and algorithms. There will be a programming component for this course as students will be required to execute, read and modify given programs, as well as write their own computer programs, based on the applications. Python is chosen as the programming language for this course as it is currently the first programming language that most students learn because of its simple syntax.    Students will continue to use and apply the foundational mathematical skills and computer science applications learned in this 200-level course, taken early in the major, in their upper-level CS courses. The computer science applications studied in this course will strengthen students’ foundations in their upper level CS courses.  The emphasis in this course is on computer science problems and applications. Mathematics topics will be taught at a level to provide a working knowledge of the computer science applications covered in this course. Computer programs will be used to demonstrate and illustrate the computer science applications and concepts, as well as to assess the students understanding of the content. Students (and instructors) must have knowledge of computer programming in Python.  The class size should be capped at 20 as this course has a lab component. | | | | |
| A.5. [Student impact](#student_impact) | After taking this course, students will have knowledge of computer science topics from programming, algorithms, compilers, networks, and cryptography. They will also learn the topics from the foundational discrete structures knowledge area in computer science. As this course will be taken early in the major, students can apply the techniques learned in this course, in later CS courses. The CS topics/applications learned here, will make students have familiarity and will be better prepared for courses such as CSCI 313 – Computer Organization and Architecture, CSCI 325 – Programming Languages, CSCI 423 – Analysis of Algorithms and other upper-level CS courses. The computer programming component in this course will strengthen the students’ programming skills. | | | | |
| A.6. [Impact on other programs](#impact) | None. | | | | |
| A.7. [Resource impact](#Resource) | [*Faculty PT & FT*](#faculty): | **This new course will not require additional CS instructors. Since CSCI 312 will be dropped, this course will replace those offerings (2 sections per year).** | | | |
| [*Library*:](#library) | **No additional resources needed** | | | |
| [*Technology*](#technology) | **Will teach in the existing computer labs using existing software** | | | |
| [*Facilities*](#facilities): | **Existing labs** | | | |
| A.8. [Semester effective](#Semester_effective) | **Fall 2021** | A.9. [Rationale if sooner than next Fall](#Semester_effective) | |  | |
| A.10. INSTRUCTIONS FOR CATALOG COPY: This single file copy must include ALL relevant pages from the college catalog, and show how the catalog will be revised. (1) Go to the “Forms and Information” page on the UCC website. Scroll down until you see the Word files for the current catalog. (2) Download ALL catalog sections relevant for this proposal, including course descriptions and/or other affected programs. (3) Place ALL relevant catalog copy into a single file. Put page breaks between sections and **delete any catalog pages not relevant for this proposal**. (4) Using the track changes function, revise the catalog pages to demonstrate what the information should look like in next year’s catalog. (5) Check the revised catalog pages against the proposal form, especially making sure that program totals are correct if adding/deleting course credits. If new copy, indicate where it should go in the catalog. If making related proposals a single catalog copy that includes all is preferred. Send catalog copy as a separate single Word file along with this form. | | | | | |

B. [NEW OR REVISED COURSES](#delete_if)  **Delete section B if the proposal does not include a new or revised course. As in section A. do not highlight but simply delete suggested options not being used. Always fill in b. 1 and B. 3 for context.**

|  | Old ([for revisions only](#Revisions)) ONLY include information that is being revised, otherwise leave blank. | New Examples are provided within some of the boxes for guidance, delete just the examples that do not apply. |
| --- | --- | --- |
| B.1. [Course prefix and number](#cours_title) |  | **CSCI 209** |
| B.2. Cross listing number if any |  |  |
| B.3. [Course title](#title) |  | **Discrete Structures Using Python** |
| B.4. [Course description](#description) |  | Students will explore computer science topics in programming, algorithms, compilers, networks, and cryptography. Fundamental mathematical concepts like finite-state machines, recurrence relations, graphs, and probability will be applied, using Python programs. |
| B.5. [Prerequisite(s)](#prereqs) |  | **CSCI 157 and Math 120, or appropriate score on placement exam** |
| B.6. [Offered](#Offered) |  | **Fall | Spring** |
| B.7. [Contact hours](#contacthours) |  | **4** |
| B.8. [Credit hours](#credits) |  | **4** |
| B.9. [Justify differences if any](#differences) |  | |
| B.10. [Grading system](#grading) |  | **Letter grade** |
| B.11. [Instructional methods](#instr_methods) |  | **Laboratory | Lecture** |
| B.11.a [Delivery Method](#instr_methods) |  | **On campus** |
| B.12.[Categories](#required) |  | **Required for major/minor** |
| B.13. Is this an Honors course? |  | **NO** |
| B.14. [General Education](#ge)  N.B. Connections must include at least 50% Standard Classroom instruction. |  | **NO** |
| B.15. [How will student performance be evaluated?](#performance) |  | **Exams | Class Work | Quizzes |**  **Projects |** |
| B.16 [Recommended class-size](#class_size" \o "Check appendix XVIII in the UCC Manual for Best Practices) |  | **20** |
| B.17. [Redundancy statement](#competing) |  |  |
| B. 18. Other changes, if any |  | |

| B.19**.** [**Course learning outcomes**](#outcomes)**: List each one in a separate row** | [**Professional Org.Standard(s)**](#standards)**, if relevant** | [**How will each outcome be measured**](#measured)**?** |
| --- | --- | --- |
| Compute with different number systems such as binary, the language of a computer. |  | Programming projects, homework and exam/quiz questions. |
| Understand how numbers are stored in computer memory, registers, or on disk |  | Programming projects, homework and exam/quiz questions. |
| Design and recognize a language of a finite-state machine |  | Programming projects, homework and exam/quiz questions. |
| Design a finite-state machine for lexical analysis used in compiler design |  | Programming projects, homework and exam/quiz questions. |
| Count the number of comparisons made in a sorting program on various input. |  | Programming projects, homework and exam/quiz questions. |
| Guess and verify an explicit formula for a recurrence relation by an induction proof to prove the software correctness of a recursive algorithm, |  | Programming projects, homework and exam/quiz questions. |
| Count the number of steps/calls in a recursive function with a given input. |  | Programming projects, homework and exam/quiz questions. |
| Apply Dijkstra’s algorithm to find the shortest path in a computer network |  | Programming projects, homework and exam/quiz questions. |
| Apply graph coloring to a scheduling problem |  | Programming projects, homework and exam/quiz questions. |
| Apply expression trees and tree traversals to understand how expressions are evaluated in a computer. |  | Programming projects, homework and exam/quiz questions. |
| Apply expression trees and tree traversals to understand how operating system processes are stored in the ready queue |  | Programming projects, homework and exam/quiz questions. |
| Understand how DFS of a tree can solve the maze problem |  | Programming projects, homework and exam/quiz questions. |
| Apply knowledge of probability to pseudo-random numbers |  | Programming projects, homework and exam/quiz questions. |
| Understand how cryptographic keys are generated by (pseudo) random-number generators |  | Programming projects, homework and exam/quiz questions. |

| B.20. [**Topical outline**](#outline)**: DO NOT INSERT WHOLE SYLLABUS, JUST A TWO-TIER TOPIC OUTLINE. Proposals that ignore this request will be returned for revision.** |
| --- |
| 1. Representation of Numbers in Computers 1 week    * Number Systems    * Binary, Octal and Hexadecimal Systems    * Largest integer represented in a computer 2. Boolean Algebra 1 week    * Propositions    * Operations on Propositions    * Predicates and Sets    * If-else statements in Python 3. Functions 1 week    * Sets and Operations    * Variables, arguments, and scope of variables    * Algorithm decomposition 4. Lexical Analysis and Parsing 2 weeks    * Finite State Machines    * Representation of FSM’s: transition diagram, transition tables    * Designing a FSM for a given language    * Recognizing the language of a given FSM    * Applications: Lexical Analysis and Parsing 5. Iteration in Algorithms 1 week    * Arithmetic and Geometric Sequences    * Computing Sums    * Sorting – Selection Sort    * Counting the steps in an algorithm 6. Recursion in Algorithms 2 weeks    * Recursive functions    * Recurrence relations    * Proof by Mathematical Induction    * Proof of Correctness of an algorithm 7. Graph Theory Applications in Networks and Routing 2 weeks    * Graph Representations: Adjacency Matrix, Adjacency lists    * Isomorphism of Graphs    * Directed and Weighted Graphs    * Dijkstra’s Algorithm – Shortest Path in a network    * Map Coloring and The Four-Color Theorem    * Application: Scheduling problems 8. Trees Applications in Search and Sorting Algorithms, and Networks 2 weeks    * BFS – Spanning Trees    * DFS and Strongly Connected Graphs    * Minimal Spanning Trees of a Graph: Prim’s and Kruskal’s Algorithms    * Expression Trees and tree traversals    * Calculating the result of an arithmetic expression on a computer    * Decision Trees as applied to sorting algorithms 9. Random number generators 1 week    * Probability    * Calculating probabilities by Counting    * Pseudo-random numbers    * Generating keys in Cryptography 10. Review and Exams 1 week   Total – 14 weeks |
|  |

## D. Signatures

* Changes that affect General Education in any way MUST be approved by ALL Deans and COGE Chair.
* Changes that directly impact more than one department/program MUST have the signatures of all relevant department chairs, program directors, and their relevant dean (e.g. when creating/revising a program using courses from other departments/programs). Check UCC manual 4.2 for further guidelines on whether the signatures need to be approval or acknowledgement.
* Proposals that do not have appropriate approval signatures will not be considered.
* Type in name of person signing and their position/affiliation.
* Send electronic files of this proposal and accompanying catalog copy to [curriculum@ric.edu](mailto:curriculum@ric.edu) and a printed signature copy of this whole form to the current Chair of UCC. Check UCC website for due dates.

##### D.1. Approvals: required from programs/departments/deans who originate the proposal. THESE may include multiple departments, e.g., for joint/interdisciplinary proposals.

| Name | Position/affiliation | [Signature](#_Signature" \o "Insert electronic signature, if available, in this column) | Date |
| --- | --- | --- | --- |
| Lisa Bain | Chair of Computer Science and Information Systems | \*approved via e-mail | 12/03/2020 |
| Earl Simson | Dean of Arts and Sciences | **Earl Simson** | 12/03/2020 |

##### D.2. [Acknowledgements](#acknowledge): REQUIRED from OTHER PROGRAMS/DEPARTMENTS (and their relevant deans if not already included above) that are IMPACTED BY THE PROPOSAL. SIGNATURE DOES NOT INDICATE APPROVAL, ONLY AWARENESS THAT THE PROPOSAL IS BEING SUBMITTED. CONCERNS SHOULD BE BROUGHT TO THE UCC COMMITTEE MEETING FOR DISCUSSION; all faculty are welcome to attend.

| Name | Position/affiliation | [Signature](#Signature_2) | Date |
| --- | --- | --- | --- |
|  |  |  |  |