Form E-1-A for Boston College Core Curriculum

Department/Program : _Computer Science

NOTE: It is only in the last year that courses in Computer Science could be used to satisfy the University Core Requirement in Mathematics. At present only one course in our program, Computer Science 1 (CSCI1101) and its Honors variant (CSCI1103) can be used for this purpose. In the Fall semester of 2019, CSCI1701, one part of a new pair of Enduring Questions courses, will be taught by the Computer Science department—nothing quite like this has been offered in the Computer Science department before. This accounts for the somewhat provisional nature of several of the responses below, especially where CSCI1701 is concerned.

1) **Have formal learning outcomes for the department's Core courses been developed? What are they?** (What specific sets of skills and knowledge does the department expect students completing its Core courses to have acquired?)

For CSCI1101 a specific set of outcomes has been developed by the group of instructors teaching the course. Attached to this document is a recent working version of these outcomes used by the instructors in 2018-2019.

CSCI1701 will be taught for the first time in the Fall of 2019. There is a preliminary version of the course syllabus, but the definitive version is still in development. Since this course is organized around a specific application area, the course content, and to some extent the associated skills and knowledge, is in part driven by the application questions.

Where are these learning outcomes published? Be specific. (Where are the department's expected learning outcomes for its Core courses accessible: on the web, in the catalog, or in your department handouts?)

The attached document concerning CSCI1101 circulated among the instructors, but some version of the desired outcomes appears in the individual syllabi for each section. A less-detailed description of learning outcomes appears in the course description posted on the department's website.

https://www.bc.edu/content/bc-web/schools/mcas/departments/computer-science/academics/courses.html

Learning outcomes for CSCI1701 will appear in the course syllabus. It is not yet known if this precise course will be taught a second time. We expect that Computer Science will continue to participate in the Enduring Questions and Complex Problems courses. While there will be a common thread to the learning outcomes, these will vary to some degree with the particular offering.

2) **Other than GPA, what data/evidence is used to determine whether students have achieved the stated outcomes for the Core requirement?** (What evidence and analytical approaches do you use to assess which of the student learning outcomes have been achieved more or less well?)

Since CSCI1101 is also a required introductory course for the major, we have tended to assess it through that lens, looking at how well it prepares students for the *next* course. The evidence here is gotten through observation of the students' work and discussion of what skills the students have acquired satisfactorily in the first course, where they appear to be deficient, etc.

3) **Who interprets the evidence? What is the process?** (Who in the department is responsible for interpreting the data and making recommendations for curriculum or assignment changes if appropriate? When does this occur?)

We hold a year-end meeting of the department where curriculum matters are discussed. For CSCI1101, the team of instructors for the course meets to discuss how the students are faring, and this often translates into recommendations for curricular change.

4) **What were the assessment results and what changes have been made as a result of using this data/evidence?** (What were the major assessment findings? Have there been any recent changes to your curriculum or program? How did the assessment data contribute to those changes?

About 6 years ago, we made a major change to CS1 by switching the language in which it was taught from Java to Python. This change was based on some years of observation by CS1 instructors that led us to conclude that Java put some significant learning obstacles for introductory-level students at the very beginning of the course. After a year or two of experience with Python, we eliminated the unit in CSCI1101 concerning object-oriented programming in Python, because we found that it crowded out other material, and tended to be confusing rather than helpful for continuing students who went on to study object-oriented programming in Java.

5) **Date of the most recent program review.** (Your latest comprehensive departmental self-study and external review.)

April, 2019.

CS 1 – Topics

0) Binary/Hex/Ascii

1) Numeric types (integer & float)

   a. Know the difference between integer and float, and how to convert between types
   b. Be familiar with arithmetic operators and expressions

2) Strings
   a. String slicing methods
   b. Other string methods: upper(), lower(), len()
3) Iteration – for loop and while loop
   a. Be familiar with for loops and while loops and the break command for "escaping" from them.
4) Conditional statements (if-elif-else statements
   a. Be familiar the relational operators: >, <, >=, <=, ==, and, !=
   b. Be familiar with how to construct complex expressions using and & or operators
   c. Be familiar with the basic if-elif-else statement, and nested conditional statements.
5) Defining function of your own
   a. You should be able to write a function and know the difference between printing the result vs using the return statement.
6) Lists
   a. Be familiar with creating lists, traversing lists, and using the **in** operator with lists.
   b. Be familiar with built-in list functions: append, remove, len. etc
   c. Know that lists are mutable, and can be updated.
   d. Basic List Comprehension
   e. Multidimensional lists
   f. Python tuples versus Python lists
7) Recursion/Iteration
   a. You should be able to define what recursion is, i.e. when a function calls itself.
   b. You should be able to write simple recursive functions (be sure to define a base case and a general case).
   c. You should be able to trace the calls of a recursive function.
   d. You should be able to write iterative and recursive versions of functions
8) Searching Algorithms (Linear Search & Binary Search)
   a. Be able to write binary search and trace the calls of the function.
9) Sorting Algorithms (Bubble/PushDown, Selection, Insertion, Merge/Sort)
10) Python Functions

General: abs bin chr float hex in int len list max min ord range raw_input randint sqrt str sum type also any math functions

Additional Functions For Strings: capitalize(), count(“”) find([“”, index]), islower(), isnumeric(), isupper(), replace(“”,””) , upper(), lower()

Additions Functions For lists: append(x), count (x), index(), insert(i,x), pop(x), remove(x), reverse(), sort(x), split(), strip()

11) Basic Exception Handling

12) Dictionary Concept and Structure

13) File Handling