William Paterson University
College of Science and Health - Department of Computer Science

Fall 2018 – Spring 2019 Assessment Cycle
Analysis of the Course Coverage and Assessment Report Data

Course Number: CS3820

Course Coordination Committee Members: John Najarian, Gilbert Ndjatou (chair)

Date: May 28, 2019

A. Course Prerequisites/Co-requisites

a) Problems/Issues Identified: None

b) Suggestions for Improvement: N/A

B. Course Objectives

a) Problems/Issues Identified: None

b) Suggestions for Improvement: N/A

C. Course Student Outcomes

a) Problems/Issues Identified:

There is a need to revise most these student outcomes because some are cannot be easily assessed or are trivial.

b) Suggestions for Improvement: see above

D. Course Content

a) Problems/Issues Identified:

A revision of the course contents is needed to include parallelism.

b) Suggestions for Improvement: see above

E. Support for the Attainment of the CS Program Student Outcomes

Student Outcome S3: Apply computer science theory and software development fundamentals to produce computing-based solutions.

In this course, students implement the language recognition functions of the tokens of a simple programming language given their specifications using regular expressions. They also implement a recursive descent parser of a simple programming language based on the CFG specifications of the syntactic structures of the language. Structure programming paradigm is also expected to be used in all programming assignments. This course therefore supports the attainment of this student outcome.

Student Outcome S5: Demonstrate abilities to locate and make effective use of information.

In this course, students are required to learn a new programming language and to produce a report based on a template provided by the instructor. They are also required to write program assignments in their chosen programming language. We then assess how substantial are their reports and the quality of their programming projects. This project necessitates a lot of research over the web and we believe that it allows us to assess the abilities of the students to locate and make effective use of information. We therefore feel that this course supports the attainment of this student outcome.

Student Outcome S7:
Demonstrate an understanding of the major programming domains and the knowledge of the most appropriate programming language for each domain.

A major objective of this course is to introduce the major programming domains and the most appropriate programming language for each domain. Questions on tests and the final exam are used to assess students understanding and their knowledge of these concepts. This course is therefore consistent with this program student outcome.

F. Analysis of the Results of the Evaluations of the Course Student Outcomes Assessment Data and Suggestions for Improvement

The results of the evaluations of the assessment data of the course student outcomes in fall 2018 are mostly good whereas those of the evaluations of course student outcomes assessment data are mostly bad in spring 2019. The instructor even notes that "this is definitely one of the most unprepared groups of students in terms of foundations in structured programming paradigms that I have taught in this course for many years." However, we need to wait some few more semesters to find out whether or not this trend continues.

G. Results of the Evaluations of the Course Student Outcomes Assessment Data

Fall 2018

Student Outcomes	Where Measured	Percentage of Satisfactory Results[1]
1. Describe the major programming domains with their characteristics and for each at least one of the most commonly used languages	Test 1, Final	73
2. Describe the characteristics of a programming language in terms of its lexical elements, and syntactic structures.	Test 1	70
3. Describe the major categories of programming languages with their characteristics.	Test 1	69
4. Describe the major methods for implementing high-level programming languages.	Test 1 final	71
5. Understand bindings and biding times	Test 2	89
6. Understand the scoping rules and lifetime of variables	Test2, final	93
7. Know the major data types of imperative programming languages.	Project	72
8. Evaluate arithmetic and logical expressions	Test 2	90
9. Use the major control structures of an imperative programming language.	Test 2, final	92
10. Understand the fundamental concepts of subprograms and parameters.	Test 2, final	80
11. Trace a program (using the different memory segments of a program) with functions' calls.	Test 2, final	75
12. To program in two or more programming language.	Project, Test 2, final	80
13. Provide the regular expressions that describe simple tokens of a programming language.	Test 3, final	73
14. Use CFG or BNF to specify simple syntactic structures of programming languages.	Test 3, final, project	71
15. Given a CFG or BNF defining a language, provide the derivation of a parse tree of a syntactic structure.	Test 3, final, project	73
16. Write a simple lexical analyzer	Project	65
17. Write the recursive descent parser of a simple programming language.	projects, test3, final	75

Notes:
[1] For the tests/quizzes/ final exam, the percentage corresponds to the number of students who receive a score of at least 70% on the question(s) related to the student outcome.

Spring 2019

Student Outcomes	Where Measured	Percentage of Satisfactory Results[1]
11. Describe the major programming domains with their characteristics and for each at least one of the most commonly used languages	Test 1, Final	53
12. Describe the characteristics of a programming language in terms of its lexical elements, and syntactic structures.	Test 1	55
13. Describe the major categories of programming languages with their characteristics.	Test 1	75
14. Describe the major methods for implementing high-level programming languages.	Test 1 final	64
15. Understand bindings and biding times	Test 2	53
16. Understand the scoping rules and lifetime of variables	Test2, final	70
17. Know the major data types of imperative programming languages.	Project	72
18. Evaluate arithmetic and logical expressions	Test 2	55
19. Use the major control structures of an imperative programming language.	Test 2, final	50
20. Understand the fundamental concepts of subprograms and parameters.	Test 2, final	56
11. Trace a program (using the different memory segments of a program) with functions' calls.	Test 2, final	65
12. To program in two or more programming language.	Project, Test 2, final	73
13. Provide the regular expressions that describe simple tokens of a programming language.	Test 3, final	67
14. Use CFG or BNF to specify simple syntactic structures of programming languages.	Test 3, final, project	68
15. Given a CFG or BNF defining a language, provide the derivation of a parse tree of a syntactic structure.	Test 3, final, project	66
16. Write a simple lexical analyzer	Project	61
17. Write the recursive descent parser of a simple programming language.	projects, test3, final	70

Notes:
[1] For the tests/quizzes/ final exam, the percentage corresponds to the number of students who receive a score of at least 70% on the question(s) related to the student outcome.

This is definitely one of the most unprepared groups of students in terms of foundation in structured programming paradigms that I have taught in this course for many years.