SYLLABUS

DATE OF LAST REVIEW:	04/2016
CIP CODE:	11.0901, 15.1201, 15.1204, 47.0104
SEMESTER:	Departmental Syllabus
COURSE TITLE:	Programming Fundamentals
COURSE NUMBER:	CIST-0120
CREDIT HOURS:	4
INSTRUCTOR:	Departmental Syllabus
OFFICE LOCATION:	Departmental Syllabus
OFFICE HOURS:	Departmental Syllabus
TELEPHONE:	913-334-1100
EMAIL:	*KCKCC issued email accounts are the official means for electronically communicating with our students.*
CO-REQUISITE(S):	CIST-0101 Computer Concepts and Applications

REQUIRED TEXT AND MATERIALS: Please check with the KCKCC bookstore, http://www.kckccbookstore.com/, for the required texts for your particular class.

COURSE DESCRIPTION:
This course is an introduction to computer programming and software development.
The goals of this course are to introduce novice programmers to the systematic and explicit design of programs and to expose students with prior programming experience to design. A multi-purpose programming language is used, but the emphasis is on designing and constructing programs and not learning a specific programming language. Extensive hands on experiences are emphasized.

METHOD OF INSTRUCTION: A variety of instructional methods may be used depending on content area. These include but are not limited to: lecture, multimedia, cooperative/collaborative learning, labs and demonstrations, projects and presentations, speeches, debates, panels, conferencing, performances, and learning experiences outside the classroom. Methodology will be selected to best meet student needs.

COURSE OUTLINE:

I. Programming with Fixed Size Data
II. Programming with Arbitrarily Sized Data
III. Abstraction
IV. Intertwined Data
V. Accumulators
VI. Change of State
VII. Universe Programming

EXPECTED LEARNER OUTCOMES:

Upon completion of this course, the student should be able to:

Overall: *The student will be able to describe and demonstrate the usage of contemporary programming methodology through designing and constructing computer programs.*

A. Develop programs that use fixed-size data like numbers, Booleans, and strings.
B. Define structured data and develop programs using it.
C. Understand when to use and write programs over structures, lists, and trees
D. Develop data models to solve programming problems
E. Write recursive and mutually recursive programs
F. Write programs that use higher order functions like map, filter, and fold.
G. Use accumulator-style programming techniques when appropriate.
H. Explain when state is needed in value-oriented programming

COURSE COMPETENCIES:

Upon successful completion of this course:

The student should be able to develop programs that use fixed-size data like numbers, Booleans, and strings.

1. Be able to convert arithmetic expression into a computer program.
2. Define a function that performs an arithmetic calculation on its inputs.
3. Demonstrate step-by-step how function calls are evaluated by Racket.
4. Write a signature and a purpose for a function.
5. Write test cases for a function using *check-expect*.
6. Write a Racket program that makes a decision using *if*.
7. "Clean up" a function by creating helper functions for common computations.

The student should be able to define structured data and develop programs using it.

8. Define compound data using *define-struct*.
9. Create examples of compound data using a constructor.
10. Extract data from a structure using a selector.

The student should be able to develop data models to solve programming problems.

11. Write a function over compound data.
12. Design and document a nested structure.
13. Write a function to process a nested structure.
14. Write a data definition for an itemization.
15. Write a template for a function over an itemization.
16. Use a template to write a function over an itemization.

17. Choose a set of good test cases for a function over an itemization.

The student should choose the proper form of data (structures, lists, and trees) when developing programs to solve problems.

18. Define a list of items using *cons* and extract pieces of a list using *first* and *rest*.
19. Use the list template to write a function over lists.
20. Go through each step in the design recipe to develop a function over lists.
21. Write functions that process a list of structs.
22. Use the function calls suggested by templates to develop helper functions for list-of-structures problems.

The student should be able to write recursive and mutually recursive programs.

23. Write the data definition for a binary tree.
24. Generate the template for a binary tree
25. Write functions over binary trees.
26. Write the data definition for a hierarchy.
27. Generate the template for a hierarchy.
28. Write functions over hierarchies.

The student should be able to write programs that use higher order functions like map, filter, and fold.

29. Develop programs using higher order functions like *map* and *filter*.

The student should be able to use accumulator-style programming techniques when appropriate.

30. Convert a Racket program to a program that accomplishes the same task "accumulator-style".

The student should be able to explain when state is needed in value-oriented programming and use it appropriately.

31. Write programs that change the values of variables.
32. Document the use of variables.
33. Document a function that changes the value of a variable
34. Choose when a problem needs to use *set!*, and when it needs to use *set-structure!*,
35. Write a function that changes a struct by using *set-structure!*.
36. Recognize when a variable is needed to record which data items have been visited.

ASSESSMENT OF LEARNER OUTCOMES:

Student progress is evaluated by means that include, but are not limited to, exams, written assignments, and class participation.

SPECIAL NOTES:

This syllabus is subject to change at the discretion of the instructor. Material included is intended to provide an outline of the course and rules that the instructor will adhere to in evaluating the student's progress. However, this syllabus is not intended to be a legal contract. Questions regarding the syllabus are welcome any time.

Kansas City Kansas Community College is committed to an appreciation of diversity with respect for the differences among the diverse groups comprising our students, faculty, and staff that is free of bigotry and discrimination. Kansas City Kansas Community College is committed to providing a multicultural education and environment that reflects and respects diversity and that seeks to increase understanding.