UNIVERSITY OF THE FRASER VALLEY

OFFICIAL UNDERGRADUATE COURSE OUTLINE (page 1)

COURSE IMPLEMENTATION DATE: May 2014
COURSE REVISED IMPLEMENTATION DATE:
COURSE TO BE REVIEWED: May 2020
(six years after UEC approval) *(month, year)*

OFFICIAL UNDERGRADUATE COURSE OUTLINE INFORMATION

Students are advised to keep course outlines in personal files for future use.
Shaded headings are subject to change at the discretion of the department – see course syllabus available from instructor

COMP 481	**Computer Information Systems**	**3**
COURSE NAME/NUMBER	FACULTY/DEPARTMENT	UFV CREDITS

Functional and Logic Programming
COURSE DESCRIPTIVE TITLE

CALENDAR DESCRIPTION:

Most programming languages (e.g. Java, C, C++, Python) are imperative languages, meaning that programs are written as sequences of instructions that change program state. However, imperative programming is just one programming paradigm. This course introduces two other programming paradigms: functional and logic programming. Logic programming is based on first-order logic, while functional programming is based on the lambda calculus. Students will learn the basic theoretical foundations as well as how to program in two relevant languages. The course will also describe the importance of these languages to the field of AI.

PREREQUISITES: COMP 251 with a C or better; MATH 225 recommended.
COREQUISITES:
PRE or COREQUISITES:

SYNONYMOUS COURSE(S):
(a) Replaces:
(b) Cross-listed with:
(c) Cannot take: for further credit.

SERVICE COURSE TO: *(department/program)*

TOTAL HOURS PER TERM: **45**

STRUCTURE OF HOURS:

Lectures:	**30**	Hrs
Seminar:		Hrs
Laboratory:	**15**	Hrs
Field experience:		Hrs
Student directed learning:		Hrs
Other (specify):		Hrs

TRAINING DAY-BASED INSTRUCTION:
Length of course:
Hours per day:

OTHER:
Maximum enrolment: **35**
Expected frequency of course offerings: **Every other year**
(every semester, annually, every other year, etc.)

WILL TRANSFER CREDIT BE REQUESTED? (lower-level courses only)	☐ Yes	☐ No
WILL TRANSFER CREDIT BE REQUESTED? (upper-level requested by department)	☐ Yes	☒ No
TRANSFER CREDIT EXISTS IN BCCAT TRANSFER GUIDE:	☐ Yes	☒ No

Course designer(s): **Gabriel Murray**

Department Head:	**Dan Harris**	Date approved:	**October 12, 2012**
Campus-Wide Consultation (CWC)		Date of meeting:	**August 30, 2013**
Curriculum Committee chair:	**David Fenske**	Date approved:	**September 20, 2013**
Dean/Associate VP:	**Lucy Lee**	Date approved:	**September 20, 2013**
Undergraduate Education Committee (UEC) approval		Date of meeting:	**October 25, 2013**

LEARNING OUTCOMES:

At the end of this course, students will be able to:

- Compare and contrast imperative and declarative languages.
- Explain the functional programming paradigm.
- Program effectively in Haskell or another functional language.
- Explain the logic programming paradigm.
- Program effectively in Prolog.
- Write programs for AI applications.

METHODS: *(Guest lecturers, presentations, online instruction, field trips, etc.)*

Lectures, labs, assignments, and guest speakers

METHODS OF OBTAINING PRIOR LEARNING ASSESSMENT RECOGNITION (PLAR):

☒ Examination(s) ☒ Portfolio assessment ☒ Interview(s)

☐ Other (specify):

☐ PLAR cannot be awarded for this course for the following reason(s):

TEXTBOOKS, REFERENCES, MATERIALS:

[Textbook selection varies by instructor. An example of texts for this course might be:]

Programming in Prolog, Clocksin and Mellish
Learn Prolog Now!, Blackburn, Bos and Striegnitz (paperback or free online)
Learn You a Haskell for Great Good!, Lipovaca (paperback or free online)
Computational Semantics with Functional Programming, van Eijck and Unger

SUPPLIES / MATERIALS:

STUDENT EVALUATION:

[An example of student evaluation for this course might be:]

Assignments:	20%
Labs:	20%
Midterm exam:	20%
Final exam:	40%

COURSE CONTENT:

[Course content varies by instructor. An example of course content might be:]

- Imperative Programming
- Declarative Programming
- Introduction to Logic Programming
- Introduction to Prolog
- Prolog for AI
- Prolog for natural language processing
- Introduction to Functional Programming
- Introduction to Haskell
- Haskell for natural language processing
- Other declarative languages