Module Code  
CS3016

Module Name  
Introduction to Functional Programming

Module Short Title  
N/a

ECTS weighting  
5

Semester  
Semester 1

Contact Hours  
Lecture hours: 22
Lab /Tutorial hours: 11
Total hours: 33

Lecturing staff: Dr Andrew Butterfield

Learning Outcomes  
On successful completion of this module students will be able to:

- Develop programs in a high level functional language
- Analyse and structure program designs in terms of functional concepts
- Understand the concept of higher-order programming inherent in functional languages
- Improve software modularity and reusability by applying higher-order principles to refactor code
- Apply a number of functional programming techniques and tools to develop effective functional systems

Module Learning Aims  
Functional programming languages present a powerful, abstract, and important direction in programming languages. The high level of abstraction and the expressive syntax makes program decomposition and composition unusually easy, while the close connections to the underlying semantics make formal reasoning tractable. Systems such as Google’s “Map/Reduce” framework demonstrate the influence of this approach, and the importance to a computer scientist of understanding it.

On this course students will learn to apply the techniques of functional programming
in a practical context. The focus is on software design and programming in the functional style, and students will “learn by doing”, through regular weekly programming assignments and case studies.

The course draws on the programming and mathematics background the students have acquired in the first two years of the degree and extends it by teaching new approaches to program design and implementation.

Course content covers both techniques and technologies. Topics will include:

- First-order functions
- Type systems for functional languages; data types; recursive types
- Lazy evaluation, infinite data structures
- Referential transparency
- Higher order functions
- I/O and State handling
- Functional debugging

Efficiency considerations

Recommended Reading List

- Richard Bird, Thinking Functionally with Haskell (Cambridge University Press, 2015, 344 pp.)

Module Pre Requisite

Some familiarity with programming would be helpful, esp. the use of program abstraction constructs such as function and procedure definitions.

Module Co Requisite

Assessment Details

% Exam: 75
% Coursework: 25
Description of assessment & assessment regulations.

An overall mark of 40% is required to pass the module. The exam and coursework components do not have to reach 40% individually. Assessment in the supplemental examinations is by 100% exam.
## Module Descriptor 2016/17
School of Computer Science and Statistics.

<table>
<thead>
<tr>
<th>Module approval date</th>
<th>N/a</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approved By</td>
<td>N/a</td>
</tr>
<tr>
<td>Academic Start Year</td>
<td>N/a</td>
</tr>
<tr>
<td>Academic Year of Data</td>
<td>N/a</td>
</tr>
</tbody>
</table>