DEPARTMENT OF
COMPUTER
SCIENCE

UNDERGRADUATE HANDBOOK SUPPLEMENT 2ND, 3RD AND 4TH Year students

Computer Science
Computer Science & Philosophy
Mathematics & Computer Science

2015

Version 1.0

Handbook Supplement for Second, Third and Fourth Year Students

This supplement is intended to give you some more information and guidance on the issues relating to your stage of the Computer Science, Computer Science & Philosophy or Mathematics & Computer Science degree.

It should be read alongside the main Course Handbook which can be found at: http://www.cs.ox.ac.uk/teaching/handbooks.html

Contents

If you have any questions regarding anything in this supplement or the handbook please speak to your tutor or contact the Academic Administrator, Shoshannah Holdom.

Disclaimer

The Examination Regulations relating to this course are available at
http://www.admin.ox.ac.uk/examregs/2015-16/hsofcompscie/studentview/

http://www.admin.ox.ac.uk/examregs/2015-16/hsomandcompscie/studentview/

http://www.admin.ox.ac.uk/examregs/2015-16/hsocscieandphil/studentview/

If there is a conflict between information in this handbook and the Examination Regulations then you should follow the Examination Regulations. If you have any concerns please contact Dr Shoshannah Holdom in the Department of Computer Science, Shoshannah.holdom@cs.ox.ac.uk

The information in this handbook is accurate as at 21st September 2015, however it may be necessary for changes to be made in certain circumstances, as explained at www.ox.ac.uk/coursechanges. If such changes are made the department will publish a new version of this handbook together with a list of the changes and students will be informed.

1 Examinations

1.1 Computer Science

Second-year Computer Science candidates will take four core courses:

- Object-Oriented Programming (OOP)
- Concurrent Programming
- Logic and Proof
- Models of Computation

Concurrent Programming, Logic and Proof and Models of Computation will each be examined by a 1.5 hour written examination.

Object-Oriented Programming will be examined by an assessed practical (35% of the marks) and a 1.5 hour written examination (65%).

Instructions for the assessed practical will be handed out on Friday in week 8 of Michaelmas Term, and the practical report must be handed in to the Examination Schools, High St., Oxford by noon on Friday of week 2 of Hilary Term. The assessed practical will incorporate and extend elements of the lab exercises that were set during term. As always, the work you submit must be your own, except where explicitly acknowledged.

Appendix A of the Course Handbook sets out the standards that are expected in this regard.

In addition, candidates offer four 1.5-hour options papers. The papers will have three questions, and you may attempt two of them. Where you are taking two papers that are time-tabled in the same three-hour session, you may work on both papers together for the full three hours, but you may not answer more than 2 questions on each paper.

In Finals papers, questions are marked out of 25. The marks for each part of each question will be indicated on the examination paper.

In the third year, candidates take six 1.5-hour option papers and submit a project report. The options are divided into three lists or schedules called B1, B2 and B4. In the third year, you may choose up to two subjects from Schedule B1 that you have not already done in the second year, up to two courses from Schedule B4, and the remainder you choose from Schedule B2.

1.2 Mathematics & Computer Science

Second year Mathematics and Computer Science students take the following Maths papers:

These courses are examined by two papers,

- A0 Linear Algebra, which will be 1.5 hours

Candidates answer two questions from a choice of three; each question is marked out of 25.

- A2 Metric Spaces and Complex Analysis

This paper includes six questions and you should answer four. The best four questions count towards a candidate's total mark for the paper.

- In addition, candidates must offer either two papers from papers A3-A5, A7-A11 or one paper from A3-A5, A7-A11 and paper ASO: A3 Rings and Modules
- A4 Integration
- A5 Topology
- A7 Numerical Analysis
- A8 Probability
- A9 Statistics
- A10 Fluids and Waves
- A11 Quantum Theory
- ASO Short Options

Candidates also take four Computer Science courses: OOP, Concurrent Programming, Models of Computation, and Logic & Proof. The OOP course will be examined by assessed practical and written paper as for Computer Science (see above).

The Computer Science papers will have three questions, and you may attempt two of them. Where you are taking two papers that are time-tabled in the same 3-hour session, you may work on both papers together for the full three hours, but you may not answer more than 2 questions on each paper.

In Finals papers, questions are marked out of 25. The marks for each part of each question will be indicated on the examination paper.

In the third year, you take a total of eight options, chosen from Schedules B1, B2, B4, and another schedule, B3, which contains Maths courses. You may choose to do between two and six options in Computer Science, and the rest in Maths. Of the Computer Science options, at most two can come from those on Schedule B1 that you have not already done in the second year, and at most two can come from Schedule B4.

1.3 Computer Science and Philosophy

Second and Third Year - Computer Science

Second year Computer Science and Philosophy students take four options from schedule A(CS&P). These four options must include Models of Computation.

Schedule A (CS&P)
Models of Computation
Algorithms & Data Structures
Compilers

Computer Graphics
Concurrency
Concurrent Programming
Databases
Intelligent Systems
Logic and Proof
Object-Oriented Programming

Object-Oriented Programming, Concurrent Programming and Logic & Proof will not be available under Schedule B1(CS&P) in your third year. If you wish to take these courses you should do so in your second year.

In the third year, you take two, four or six Computer Science courses. Options are chosen from Schedules B1(CS&P), B2(CS&P) and B4(CS&P). You can choose at most two subjects from Schedule B1(CS&P) that you have not already done in the second year and at most two from Schedule B4(CS&P).

Schedule B1 (CS&P)
Algorithms & Data Structures
Compilers
Computer Graphics
Concurrency
Databases

Schedule B2 (CS&P)
Computational Learning Theory
Knowledge Representation and Reasoning
Computational Complexity
Computer-Aided Formal Verification
Computers in Society
Computer Security
Intelligent Systems
Lambda Calculus and Types
Principles of Programming Languages

Schedule B4 (CS&P)
Communication Theory (B11a)

Second and Third Year - Philosophy

You will take three, four or five Philosophy courses during your second and third years, from the following list of courses. It is recommended that you take two courses in your second year.

101. History of Philosophy from Descartes to Kant; 102. Knowledge and Reality; 103. Ethics; 104. Philosophy of Mind; 106. Philosophy of Science and Social Science; 107. Philosophy of Religion; 108. The Philosophy of Logic and Language; 109. Aesthetics; 110. Medieval Philosophy:Aquinas; 111. Medieval Philosophy: Duns Scotus and Ockham; 112. The Philosophy of Kant; 113. Post-Kantian Philosophy; 114. Theory of Politics; 115. Plato, Republic; 116. Aristotle, Nicomachean Ethics; 117. Frege, Russell, and Wittgenstein; 118. The Later Philosophy of Wittgenstein; 120. Intermediate Philosophy of Physics; 122. Philosophy of Mathematics; 124. Philosophy of Science; 125. Philosophy of Cognitive Science; 127. Philosophical Logic.

You must include at least two of 101, 102, 104, 108, 122, 124, 125 and 127. Full course details can be found on the Philosophy website at:
www.philosophy.ox.ac.uk/undergraduate/course_descriptions

Note that each Philosophy option is twice the weight of a Computer Science option.

1.4 Composition of fourth year papers

You will have the option of continuing for a fourth year, if you have achieved at least upper second class Honours in the second and third years together.

In the fourth year of Computer Science you are required to take five courses and a Computer Science project. The courses are chosen from a schedule called C1, which is published at http://www.cs.ox.ac.uk/teaching/bacompsci/PartC/

In the fourth year of Mathematics and Computer Science you are required to take either five courses and a Computer Science project *or* six courses and a Mathematics dissertation. The courses are chosen from Schedule C1 and Schedule, C2. There is no restriction on the number of courses chosen from each schedule. The schedules are published at http://www.cs.ox.ac.uk/teaching/mcs/PartC/. Note that if you choose to submit a Mathematics dissertation, you must also choose at least two other Mathematics courses.

In the fourth year of Computer Science and Philosophy, you may take courses according to the following rules:

- Each Philosophy paper or thesis is worth 8 units;
- Each Computer Science taught course is worth 3 units;
- A Computer Science project is worth 9 units.

You must complete between 24 and 26 units, subject to the following constraints:

- You may take at most six Computer Science taught courses;
- You may not take both a Philosophy thesis and a Computer Science project.

Computer Science courses are chosen from Schedule C1. Philosophy courses are chosen from courses 101-120, 122, 124, 125, 127 and 180, as described on the Philosophy Faculty website. Each Philosophy course will be assessed by a 3-hour written examination together with an essay of at most 5,000 words.

Rules for Philosophy theses are described in the Grey Book except that the word limit is 20,000 words. More advice on Philosophy essays and theses will be issued later in the year.

The effect of these rules is that you should take one of the following combinations:

- Three Philosophy papers (maybe including a thesis) (24 units);
- Two Philosophy papers (maybe including a thesis) and either three CS courses or a CS project (25 units);
- One Philosophy paper (or thesis), and six CS courses (26 units);
- One Philosophy paper, three CS courses and a CS project (26 units);
- Five CS courses and a CS project (24 units).

Each option will be examined either by a sit-down paper, which is likely to be of two hours' duration, or by a mini-project (please note that Computer Science options will be examined by mini-project, with the exception of Probability and Computing which will be examined by a sit-down paper).

Computer Science mini-projects will be handed out on the last Friday of the term in which the subject is being taught, or for subjects shared with the MFoCS course Monday of week 8 of the

term in which the subject is being taught. This information will be included in the Notice to Candidates sent out each term.

Mini-projects must be handed in to the Exam Schools by noon on Monday of week 0 of the following term. The mini-project will be designed to be completed in about three days. It will include some questions that are more open-ended than those on a standard sit-down exam. The work you submit should be entirely your own work. If you make use of material from web-sites, books, articles or other sources you must acknowledge these and give suitable references. **Please see the Appendix on plagiarism in the Course Handbook.**

Although you will be doing examinations at the end of each term, you will be entering for these exams in the normal way (i.e. Friday of Week 2, Hilary Term). You must make sure you enter for the examinations that you took in Michaelmas Term.

Please note that the Computer Science courses in Part C are 50% bigger than those in earlier years, i.e. for each course in the 3rd year undergraduates are expected to undertake about 10 hours of study per week, but 4th year courses will each require about 15 hours a week of study. Computer Science lecturers are providing this extra work in a variety of ways, e.g. some will give 16 lectures but will require you to undertake extra reading, classes and/or practicals, whereas others will be giving 24 lectures, and others still will be doing something in between. Please look at each synopsis for details on this.

1.5 Important Dates

Dates of Term: 2015-16

Michaelmas Term:	Sunday 11th October 2015 – Saturday 5th December 2015
Hilary Term:	Sunday 17th January 2016 – Saturday 12th March 2016
Trinity Term:	Sunday 24th April 2016 – Saturday 18th June 2016

Practicals (All years and all programmes of study)

- Friday of 5th week of Trinity Term by 12 Noon (to the Department)

2nd Year – Object Oriented Programming Practical Assignment

- Friday of 2nd week of Hilary Term by 12 Noon (to the Examination Schools)

2nd Year Group Design Practical

Final Report – Usually by the end of Week 2 of Trinity Term
Presentation – Usually in Week 3 of Trinity Term

3rd Year Computer Science Project Report

- Monday of 5th Week of Trinity Term, by 12 Noon (to Examination Schools)

4th Year Computer Science Project Report

- Monday of 5th Week of Trinity Term, by 12 Noon (to Examination Schools)

2 Practicals

Many courses in the second, third and fourth years have associated practical work, and this forms a compulsory part of those courses. Arrangements for these practicals are described in the Course Handbook. A good performance across all practicals can result in being awarded a distinction in practical work which is noted on your degree transcript.

2.1 Group Design Practical

The second year course also includes a group design practical as part of the practical requirements for the year. This will allow you to practise the skills you learnt in the core programming courses, and to begin to develop a range of further skills including team-working, project and time management, and presentation skills.

The group design practical is intended to take you 20-30 hours, mainly during Hilary Term. There will be a briefing meeting early in Hilary Term, setting out the aims and format of the exercise and listing several possible problems to tackle. You will then be allocated to a team of around 5 people to work on one particular problem together. Each team will be allocated a member of staff to act as a supervisor, and will have three one-hour meetings with their supervisor during the project.

The first meeting with the supervisor will take place at the beginning of Hilary Term, where the group will present a specification and project plan.

The second meeting with the supervisor will take place in Hilary Term: the group will present their initial module implementations and test results.

The third meeting will take place in Trinity Term: the group will demonstrate their product and deliver a brief final report. Each student will also deliver to the supervisor a one-page summary of their individual contribution.

Finally, the groups will present their work to students, members of the Department, and guests. This will take the form of a demonstration session, followed by a seminar where groups will take turns to describe their projects; and prizes will be presented.

The final group report and summary of individual contribution will be assessed as S+, S, Pass or Fail. The group design practical counts as one-third of the total practical mark for the second year and candidates are required to achieve at least a Pass. Your supervisor will submit your group report and your summary of your individual contribution to the Examiners to be considered along with your other practical reports.

3 Projects

As described in the *Examination Regulations*, undergraduates in the third and fourth years of the Final Honour School of Computer Science are required to undertake a project. Mathematics & Computer Science undergraduates are required to undertake a Computer Science project or a Mathematics dissertation in their Fourth Year. These informal notes are intended to supplement, but not replace, the formal regulations; we hope they will be useful to you.

3.1 Amount of work

The project amounts to about ***one third*** of your work in the third year, and likewise in the fourth year, and so should be thought of as being *about a whole term's work*. When exactly this work is done should be a matter for discussion with your tutor and/or project supervisor, but it is recommended that you make a solid start during the long vacation at the end of the previous year, and aim to finish by Easter.

Fourth year students are required to submit an initial report on the background to their project by the end of Michaelmas Term. Your supervisor should discuss the report with you and ensure that it is suitable before you are allowed to continue with the project.

3.2 Choosing a project

You should begin by discussing the project with your tutor. Projects might involve the specification, design and implementation of a piece of software or hardware; the further development of some theoretical ideas, or the use of existing computing tools to perform some analysis. They might be directly related to material in the more advanced computing courses, although this will not always be the case.

A list of project outlines proposed by potential supervisors will be circulated; projects need not be drawn from this list, but this may serve both as a guide to the drawing up of proposals and as help in finding supervisors. You are free to construct your own project proposal, in consultation with your tutor and a potential supervisor – who may or may not be your tutor. Supervisors are normally expected to be a member of the Faculties of Computer Science, Mathematics or Engineering Sciences or a person of similar seniority in the Department of Computer Science.

Whether you write your own proposal, or propose to undertake a project from the circulated list, *you need to obtain the permission of the person you propose as supervisor before you submit your proposal*.

Project proposals must be approved by the Undergraduate Supervisory Committee of the Department of Computer Science. Normally the committee will devolve responsibility for this checking and approval to informal meetings of its Projects Committee.

It would be unwise to spend too much time on an unapproved project. Therefore it should be the normal practice to submit proposals by *end of the Easter vacation in the year before you undertake the project*. There is nothing to stop you submitting a proposal earlier than that and the committee would welcome your doing so. Proposals should be delivered to the Secretary to the Academic Administrator at the Department of Computer Science, Wolfson Building, Parks Road.

If you have decided on a project, but failed to identify a supervisor, the project co-ordinator will try to find you an appropriate supervisor during Trinity Term. Students who have not decided on a project, or found a suitable supervisor, will be allocated a project and supervisor during the summer vacation. It is in your best interests to avoid this by submitting a proposal in good time.

3.3 Carrying out the project

During the project you should expect to meet your project supervisor for about half an hour per week, on average; however, this figure might vary, depending on the nature of the project. You should be prepared to take the initiative in arranging meetings with your supervisor.

Context and Scope

To decide on the exact scope of your project you first need to investigate the background and context of the area you are working on. Your project should address a well-chosen set of concerns that are appropriate to this context. Ideally, you should identify a small number of more difficult problems, and use your project as a vehicle to explore solutions to them.

For example, a program that allows human players to compete with each other in playing a game over a network might present a number of significant problems. The play will take place over a network that might be unreliable, or one or other player might quit the game before it is finished, and it would be important for the program not to become stuck if one of these things happened. For a multi-player game, players might be able to join or leave at different times, and it would be important that the program would continue to function. In a large game, it might become important to minimize the number of direct communication links that were used, and yet still have the program be robust to failure of computers or network links. Also, players may be able to act concurrently, and the outcome of concurrent activity must be determined by the program in an accurate and fair way.

It would not be necessary to address all these aspects in a project, but it would be good to show an awareness of most of them and concentrate on some of them in your implementation and testing. It would be a mistake to devote too much effort to polishing the GUI at the expense of addressing the more fundamental networking and concurrency issues.

Undergraduate projects are not expected to be original in the same way as research papers, and a perfectly reasonable project would be to take some ideas from a published source and create your own implementation of them. For a good project, there must however be some element of original contribution. For example, there is a book that describes in detail the techniques that are generally used in implementing computer chess programs, and it would not be sufficient for a first class project just to implement some ideas from the book; instead, there would have to be a significant amount of testing, evaluation and assessment of the different methods chosen.

Choice of Technology

The project should be implemented using appropriate technology. The report should contain a brief explanation of why the technical solution (such as a programming language and libraries) was chosen, but it's not necessary to give a long comparison of alternative approaches.

It is good to choose a technological basis that makes the programming easier, allowing you to concentrate on distinctive aspects of your problem area. For example, you could use a networking library that supports transmission of structured values instead of building the same functionality from scratch, and that would be sensible if the point of the project is some higher-level application. Or you could use a functional programming language to make a prototype of a compiler and abstract machine and avoid the heavy work of implementing these in low-level code.

3.4 Writing the report

Your report is the only way that your achievement is communicated to the examiners. Its writing should therefore be treated as a substantial part of the work involved and a suitable amount of the time should be allocated to it – perhaps a fifth.

It is a very good idea to write the report as you go along: it is far easier to describe things when they are still fresh in your mind. Of course, your ideas will develop as the project proceeds, so you will have to go back and revise material at the end.

Structure and contents

The Examination Regulations do not lay down a format for the report. It should be considered to be a technical document designed to be readable by a computer scientist who is not a specialist in the topic, say one of your colleagues.

The sort of structure that would suit many programming projects is as follows:

- Abstract: a brief description of what you did; no more than a page.
- Contents.
- Introduction: a description of the objectives of the project, and what makes them worthwhile; an overview of the achievements; a road map of the report.
- Background: any background information that the reader will need to know (and is unlikely to already have) to understand the rest of the report.
- Requirements: a description of the requirements of the program.
- Design: maybe including: a description of how you broke the problem down, perhaps supported by class diagrams and/or sequence diagrams; a description of any interesting algorithms or data structures that you used; a description of the user interface, perhaps supported by screen shots. Note that UML class diagrams on their own are usually not sufficient to explain a design.

  You should make it clear why the design of your program can be expected to solve the problem. You might like to discuss alternative designs that you considered, and why you decided that they were less appropriate than the one you chose.
- Testing and evaluation: describing what strategy you used to test the program or evaluate its usability or performance, and how the results compared with those that were expected.
- Conclusions: a summary of your achievements; a critical appraisal, describing what worked well and what could be improved; a discussion of the lessons you have learnt from the project.
- References: giving author, title, and publication details of works to which you referred.
- Appendices: supporting material not needed in the text. It is not required that you include a listing of the program you have written, though many reports do so. Any code that you do include in an appendix must be well laid-out, properly divided into manageable modules and subroutines with well-chosen names, and provided with appropriate comments. Note that the body of the report should be comprehensible without reading the appendices.

If you have carried out a more experimental project, then you should discuss how you designed the experiment(s), your results, and the conclusions you draw from them.

A good project will always contain a significant amount of *analysis and assessment* of the results that have been obtained. It's not enough to produce a program, say that you've tried it a couple of times, and that it seems to work! The analysis and assessment component of a project may take many forms, depending on the nature of the project itself. For programming projects it may take the form of a systematic approach to testing, a careful analysis of competing design choices, or a thorough evaluation of the effectiveness of the program for the problem it was created to solve.

It is appropriate for supervisors to read and comment on a draft of the report, and to offer advice on suitable references and methods. It is also possible for the work reported upon to be a part of a piece of work being undertaken by several people, but the contribution of the individual project must be clearly identifiable, and clearly explained in the report. The report must be the work of the candidate alone (except for any clearly identified common material in joint projects). **Please see the Appendix on plagiarism at the end of this supplement.**

Submitting the Report

The *Examination Regulations* include bounds on the length of the report: it should not exceed 10,000 words and 40 pages of additional material. Note that these figures are *limits*, not targets.

Two copies of the report are required. It should be legible, and you will be expected to type it. (The regulations may not seem to be clear about this, but 'typed' is intended to allow 'LaTeXed', or otherwise word-processed.) The copies of the report which you hand in should be securely bound. This does not mean that you need to go to a bookbinder's: any of the various ways of securing the pages of a short document to each other will do, but please make sure that you do not hand in something which can fall apart.

Your supervisor will be asked by examiners to answer a questionnaire about the project and the amount of help they gave. This questionnaire helps the examiners to satisfy themselves that the project is your own work, and to assess the contribution you made in carrying it out. Your supervisor will want to be able to report to the examiners that the software is working properly, and for this purpose you should make sure that (s)he sees a demonstration of it in action towards the closing stages of the project. It is up to you to agree with him or her when this takes place and what form of demonstration is appropriate to the kind of software you have developed. There is no need to arrange a separate, formal demonstration if the supervisor has seen the software in action over the course of its development.

Note that it is pointless to include a CD with your project submission, or to refer to a web page where the project results can be found. The assessors cannot be expected to take the time to load the software from all projects, and since a CD is not a required part of your submission, it would be wrong of them to give you extra credit for what it contains, compared to someone else who submitted a clear report with no CD.

Style

The aim of the report is to communicate what you have done to a reader who is not an expert in the area of your project. Good style will help you; poor style will obscure your intent. You should therefore put effort into presenting your ideas as clearly as possible. If you do not have much experience of writing, then it is a very good idea to read a book or article on the subject, such as

. The Elements of Style, by William Strunk, Jr, and E.B. White;

. Clarity in Technical Reporting by S. Katzoff, NASA

Think about what you want to say, and the order in which you want to say it: try to identify a clear flow of ideas.

Make your points in a clear and unambiguous manner. Use diagrams and graphs where they enhance or clarify the text.

Help the reader as much as you can. Insert sentences at the beginning of sections to summarise the main ideas: that way, the reader knows what to look out for. Insert sentences at the end of sections to reiterate the main ideas: that way, the reader can be sure (s)he has picked up the main points. If you include some mathematics, you should also include some words describing its meaning. Include examples to help illustrate difficult ideas.

Try not to include irrelevant material to pad out your report. This may result in the reader becoming bored and losing concentration. On the other hand, it is important to include small details which are essential to the reader's understanding.

Your supervisor will read your draft text and help you with its content, accuracy and style. But you should make any draft as good as you know how before showing it to your supervisor. Your implementation of the advice you get is your own responsibility. You should not expect your supervisor to read your final draft: that is the task of the examiner.

Originality

It is of the utmost importance that your project report is your own work. Whenever you include a quotation, or paraphrase of the work of others, you should make this clear by giving a reference. Direct quotations should be within quotation marks, or indented. Such direct quotations should be rare, such as where you want to discuss another writer's opinion. Do not be tempted to construct large sections directly from sources: the examiners want to see evidence that you understand the material, not just that you are able to use Google and cut and paste. Your project supervisor will be able to give you advice about the style and format of references, and the extent to which they are needed when you describe the background to your work. *Sources which must be acknowledged include all materials that have been written by others, whether printed or electronic, published or unpublished.* **Please see the Appendix on plagiarism at the end of this supplement.**

The Proctors' and Assessor's Memorandum contains explicit warnings about the consequences of plagiarism, that is, of failing to give due credit to the work of others in material that is submitted for examination. Those found guilty of plagiarism may be severely penalized: penalties include fines, deduction of marks, or outright failure of your degree.

Declaration of Authorship

When you submit your project report you are also required to submit a declaration of authorship. You will need to complete this form to declare that the work you are submitting is your own, except where stated otherwise, and to include it in the envelope with your report when you submit it to the Examination Schools.

Assessment

Projects are marked on a scale from 0 to 100. However, in practice, very few projects are given a mark that is higher than 80 or lower than 50, so that the marks can be combined with marks for written papers without excessive scaling.

First class (70-80):	A complete project that addresses a well-rounded collection of relevant *concerns*, using appropriate *technology*, shows some aspects of *originality*, involves a significant amount of *analysis* or *assessment* of results, and is written up in a *clear report*.
Upper second class (60-70):	A basically complete project that achieves most of its aims, but does not address some of the appropriate concerns, or follows an obvious implementation path, or has not been thoroughly tested or assessed, or is written up in a less clear report.
Lower second class (50-60):	An incomplete project that may represent a start on a feasible plan, but leaves substantial parts still to be completed. Alternatively, a project that fails to address many of the appropriate concerns, or is far too unambitious, lacks any analysis, or is very unclear.
Third class (40-50):	A very incomplete project, perhaps with fragments only of a program, and a plan that remains vague. Alternatively, a project that shows poor understanding of the relevant area, or contains serious errors.
Marks below 40:	Marks below 40 may be awarded for very insubstantial reports indicating little serious engagement with the material.

To arrive at these marks, the assessors are asked to consider the following questions:

- **Context**: does the report show a good appreciation of the context to the work, giving suitable motivation, relevant background and appropriate references?
- **Competence**: does the report demonstrate competence in the use of appropriate techniques, tools or technology at a suitable level of expertise?
- **Contribution**: does the report show that the student has made some original contribution to the topic, designing and implementing an appropriate system?
- **Criticism**: does the report provide appropriate critical assessment and evaluation of the work that has been done, and the process of doing it?
- **Clarity**: is the report written in a way that is readable and clear for the non-specialist, but with appropriate level of detail to document the work done?

4 Three or Four Years?

As you know, the Department of Computer Science offers both three or four year degrees: the three year degrees lead to the award of a BA in Computer Science, BA in Computer Science and Philosophy or a BA in Mathematics and Computer Science., while four year degrees lead to the awards of Master of Computer Science, Master of Computer Science and Philosophy or Master of Mathematics and Computer Science.

The four year degrees give you a chance to follow advanced, research-oriented courses, taught by world experts in the field. See the departmental website for more details. The four year degrees give ideal preparation for a research degree, or a research-related job in industry.

You will need to decide early in your third year whether you intend to carry on for a fourth year. You should discuss your choice with your tutor. Your third year examination entry will have a space for you to indicate if you are staying on for the fourth year, this will be given to you by your college in Hilary Term. In order to continue into a fourth year you need to be awarded at least a 2:1 at the end of your third year.

5 What next?

5.1 Higher degrees

Many of our graduates go on to do a higher degree – an MSc or a PhD or DPhil – at Oxford or elsewhere; perhaps that interests you.

An MSc may be the appropriate path for you if you expect to get at least a II.i in Finals and wish to specialise. In most MSc courses a project plays an important part; for that, evidence of academic motivation and independence of thought are important. There is a variety of interesting MSc courses around the country, many of which specialise in topics for which you may be well prepared, depending on your choice of subjects; they range from parallel processing, artificial intelligence, through computational statistics to numerical modelling. Your tutor will be happy to advise you.

If you expect to get a First in Finals you may be interested in doing a DPhil. It is important that you realise that a DPhil is not awarded simply for three years of programming. Whilst being adept at programming, you should also have a strong command of the theory and the relationship between the two. As an undergraduate you should have attempted not just the routine tutorial problems, but have demonstrated some creativity and ability to solve harder problems. You should have a critical outlook with strong motivation and independence of thought, and above all a desire to reflect on what you have produced, incorporating the result of your reflection into your work. Typically, you should hope to produce a thesis which makes some novel theoretical contribution and shows how it can be usefully applied.

Talk to DPhil students in the department; discuss the prospect with your tutor if you think you might be interested.

It is worth talking to potential supervisors early (ideally before the end of your penultimate year). This might give them time to find money to fund you!

To apply: in Michaelmas Term of your final year, obtain application forms from: www.ox.ac.uk/admissions/graduate

You will need references from two or three referees; it is usual to choose tutors, project supervisors and college lecturers.

If you have questions about graduate study in the Department of Computer Science please contact the Graduate Studies Administrator, Mrs Julie Sheppard in room 112 or by email julie.sheppard@cs.ox.ac.uk,

5.2 Careers

Information about careers is provided by Oxford University Careers Service, 56 Banbury Road. The Careers Service organise many events to help you choose a career that suits you, and to put you in touch with recruiters. Their web site is at: www.careers.ox.ac.uk

You are urged to contact the Careers Service for detailed information on careers, and also for advice on compiling a CV, on how to apply, and on interview technique.

When we receive information about careers suitable for Computer Science graduates, it is put on the Careers notice board in the basement of the Department of Computer Science or circulated by email. Information on job vacancies (together with summer internships and competitions) can also be found on our web site at www.cs.ox.ac.uk/industry/internal/vacancies.jsp (NB this site can only be accessed from within the Oxford domain).