

---

**Department of Computer Science, Australian National University**  
**COMP1100 — Introduction to Programming and Algorithms**  
**Semester 1, 2006**

**Assignment 2 — Grade Roster**  
**Due: 7.00pm on Friday 12th May, 2006**

---

### Outline

The aim of this assignment is to give you more experience with designing and implementing a relatively simple program involving several modules and structured data representing the problem domain.

This assignment comprises 15% of your total assessment for COMP1100 *Introduction to Programming and Algorithms*.

### Task Description

The calculation of grades in a university course depends on a number of assessment items, including tutorial participation, assignment marks and exam marks. The aim of this assignment is to carry out such calculations for a fictitious class and assessment scheme.

The input to the program is a file containing the assessment data for each student, organised in a set format. For example, here is a small extract:

```
Adrian Belew
Tutes: 1 1 0 1 1 0 1 1 0 1
Assts: 7.5 10 8
Final: 80
```

```
Terry Bozzio
Tutes: 1 1 1 1 1 1 0 0 0 0
Assts: 9.5 10 4.5
Final: 35
```

The data for each student is over 4 lines, beginning with the student's name.

- The second line is their tutorial attendance record. There are 10 tutorial marks, with 1 indicating that the student attended that tutorial and 0 indicating that they were absent. Tutorial participation contributes 10% to a student's final mark (1% for each tutorial attended).
- The third line is the marks for their three assignments. Each assignment is marked out of 10 (fractional marks are possible) and each contributes 10% to a student's final mark.
- The fourth line is their final examination mark as a percentage. As you may infer, the final exam counts as 60% of a student's final mark.

The output is a *Grade Roster*, being a list of all students, their percentage marks and their final grade. For example:

Jimmy Carl Black	70	D
Adrian Belew	81	HD
Terry Bozzio	52	P
Napoleon Murphy Brock	61	CR
Ray Collins	48	N
Suzy Creamcheese	100	HD
Steve Vai	0	N

Final percentage marks are *rounded up* to the nearest whole number. Students achieving a final mark of 80% or better are awarded a High Distinction (*HD*); 70% or better is a Distinction (*D*); 60% or better is a Credit (*CR*); 50% or better is a Pass (*P*); marks below 50% are a Fail (*N*).

Your task is to perform the calculations to determine each student's final mark and grade and to produce a grade roster presented in a similar format to the example above. You may find useful the `fromIntegral` function in the standard prelude, which converts values of type `Int` (or `Integer`) to any other `Num` type, including `Float`.

To get you started, I have defined several data structures appropriate for representing the problem. These are in the modules `StudentRecord.hs` and `Results.hs` available from the COMP1100 web site and at `/dept/dcs/comp1100/public/assts/asst2/`. There is also a main module, `Main.hs`, which includes most of what you will need in the `main` function. In particular, I have given you the code to read an input file of assessment data and produce a data structure of type `Class`, representing the assessment components of each student in the class. You will need to extend the modules I have provided, and develop some other modules of your own design.

Since a primary focus of this exercise is the structure and modularisation of programs, you should carefully consider the organisation of your program into an appropriate collection of modules, building on those already provided. Carry out this important design activity *before* you start any code development.

Successful completion of the tasks outlined above will make it possible for you to achieve a Credit grade on this assignment. Note that it does not *guarantee* you a Credit grade — it only fixes an upper bound.

**Grade Summary:** To be eligible for a *Distinction* grade on this assignment, your program output should also present a summary of the grades awarded to the class. For example, the following summary should follow the example grade roster shown above:

Grades Summary		
-----		
HD:	2	(29%)
D:	1	(14%)
CR:	1	(14%)
P:	1	(14%)
N:	2	(29%)

The grades summary gives a count of the number of students awarded each grade, together with the percentage of the class awarded that grade, rounded to the nearest whole number.

**Statistical Analysis:** To be eligible for full marks on this assignment, you should also produce a statistical analysis of the assessment items for the course. In particular, you should report the average mark for the final exam, each of the assignments, and the course as a whole. You should also report the attendance at each tutorial as a raw count and as a percentage of the class.

If you're so inclined and have time to spare, you may wish to consider producing a bar graph of final marks, perhaps using the ANUPlot Graphics library. We will be impressed, and you will learn something, but no marks will be allocated to this activity.

## Submission

You should submit all the modules that you have written and wish to be included in your assessment, by the published deadline. Late submissions will be penalised 10% for each day (or part thereof) past the deadline.

Assignment submission will be electronic. Submit your assignment with the command:

```
submit comp1100 Asst2 <file names>
```

Only submit plain text files. Unless you have a very good reason (explained in a submitted README file) the only files to submit will be Haskell modules (with `.hs` suffix). Do not submit anything in `tar`, `zip`, or any other such format.

Make sure you practice submitting your files well before the deadline. It is possible and perfectly acceptable to submit updated versions of your files. Our system will collect them all and your most recent submission will be the one assessed.

You should check that your submissions have been successful by clicking the **View Marks** button in **StReaMS** (<http://cs.anu.edu.au/streams/>). You should see a list of files together with the time and date of submission.