CSE 1301 PRACTICAL SESSION 12
Sorting and List Manipulation (10 marks)

The aim of this prac is to give you practice at writing code for sorting and manipulating lists.

Coding style and documentation: You are expected to document your programs and to use a standard, clear, and consistent coding style. Up to 2 of the marks for each practical class may be deducted for poor coding style and/or inadequate documentation.

Preparation (to be completed before class) (2 marks)

Write all the algorithms required for the questions below, and attempt the code for all the questions.

Supplied information: Handling student and class data

This prac requires you to use the same code for handling student and class data that you used for Prac Session 11, available from

http://www.csse.monash.edu.au/courseware/cse1301/pracs/prac11_code

PART 1: Sorting (4 marks)

(a) The records in the student file are generally unsorted, but some of the records appear in sorted order. Select a sorting procedure which takes advantage of this fact, and *justify your selection*. Modify the C code of this procedure so that it receives the following parameters
- a *sorting key* (which could be any of the members of the student struct)
- the class struct.

and sorts the students in the class according to this key. For example, if the sorting key is "last name", the students should be sorted according to their last name.

The code for the three sorting functions is available at
http://www.csse.monash.edu.au/courseware/cse1301/lectures/lect32/lect32_examples/

Note that you are not allowed to use the quicksort methods from the C library. You must use one of the sorting procedures provided above.

Hint: You may want to use the library function `sprintf`, which prints into a string instead of to `stdout`. This will allow you to write a function that can compare any pair of key values regardless of their type using `strcmp`. You can assume all ID numbers to have no more than 8 digits and all marks to be in the range 1 to 100.

(b) Write a C program that reads a file of student records into the class struct and then calls your sorting function with each of the possible sorting keys. After each sort operation, your program should print the sorted class list.

Note: You should use the supplied functions for reading the file and printing class details.

PART 2: List Manipulation **(4 marks)**

(a) Modify the function `addElement` presented in class, so that it adds an element to the array in ascending order of the student's last name and first name. Your function should handle the following cases
   i. the list is empty
   ii. the list is full
   iii. the element to be inserted is already in the list (in which case, the new element is not added).

(b) Write an algorithm for a function `deleteElement`, which removes a student with a given last name and first name from the list. Your function should handle the following cases:
   - the list is empty
   - the list is full
   - the element to be deleted is not in the list (in which case, the element cannot be deleted).

   Note: you can use sequential search or binary search to find a student, and you can reuse your search code form Prac 11.

   Code, test and document your program.

Submission:

Preparation: Show written preparation work to demonstrator at start of class for marking . If not done before class, it will not receive any marks, however you should still show it to your demonstrator during class to check your understanding.

PART 1, 2: To be marked by the demonstrator during this class. No late submission.