THE UNIVERSITY OF WAIKATO

COMPUTER SCIENCE SCHOLARSHIP EXAMINATION 2009

PRACTICAL SECTION

TIME ALLOWED:	Six hours with a break for lunch at the discretion of the supervisor
NUMBER OF QUESTIONS IN PAPER:	Three
NUMBER OF QUESTIONS TO BE ANSWERED:	Three
GENERAL INSTRUCTIONS:	Candidates are to answer ALL THREE questions. All questions are important. Answer as much of each question as you can. Plan your time to allow a good attempt at each question, but be aware that Question 3 is the most difficult and will take considerably longer than the others.
SPECIAL INSTRUCTIONS:	Please hand in listings, notes and answers to written questions, and a CD/floppy disk with your program/computer work for each question. Please make sure that copies of programs are stored as plain text files. You cannot assume that the examiner has available any special software that might be required to read your files. Candidates may use any texts or manuals for reference during the examination.
CALCULATORS PERMITTED:	Yes

1. **Generating Damage (Problem solving using a spreadsheet)**

In this question you are asked to use a spreadsheet to do calculations and to display the results. We expect that the spreadsheet will be used for all calculations unless the question states otherwise - you will be marked down for performing calculations by hand and directly entering the results. Your work will be graded on three criteria.

(a) The accuracy of your results.

(b) The skill you show in making use of the capabilities of the spreadsheet.

(c) The presentation of your results. We have deliberately not provided any instructions concerning layout or formatting and example graphs may lack labels and proper scales.

The task of generating power for a country is a difficult one. Worrying about the financial aspects and planning for good reliability of supply are hard enough, but it is also important for power companies to try to generate energy in the least environmentally damaging way. In this exercise you are presented with some information about power consumption and data on the environmental impact of different power stations. Your task is to help planners by displaying information on overall environmental impact of particular strategies of power station use.

Your country has 30 power stations, numbered 1 to 30. For each power station you are given the capacity of the station in megawatts, and an estimate of the environmental damage it is doing per megawatt of power generated.

Station Number	Capacity (megawatts)	Environmental Damage / MW
1	700	0
2	640	0
3	960	0
4	230	0
5	240	0
6	720	0
7	140	0
8	190	0
9	710	10
10	500	10
11	830	20
12	360	30
13	750	30
14	260	30
15	150	30

Station Number	Capacity (megawatts)	Environmental Damage / MW
16	740	30
17	470	40
18	530	50
19	250	50
20	800	60
21	730	70
22	580	70
23	860	70
24	780	70
25	240	70
26	160	80
27	870	80
28	840	100
29	100	100
30	210	100

The Environmental Damage is just a number between 0 and 100, where 0 represents very low damage (possibly a hydro station) and 100 represents high damage (possibly a coal burning station). Each station can be operated at any level of power output from 0 up to its capacity. It only does environmental damage when running. The rate at which it does damage is its damage level multiplied by its power output. For example station 30 has maximum capacity of 210 MW and damage rate value of 100; so when it is generating 70MW (only ⅓ of its maximum) it is doing damage at a rate of 70 x 100 = 7000.

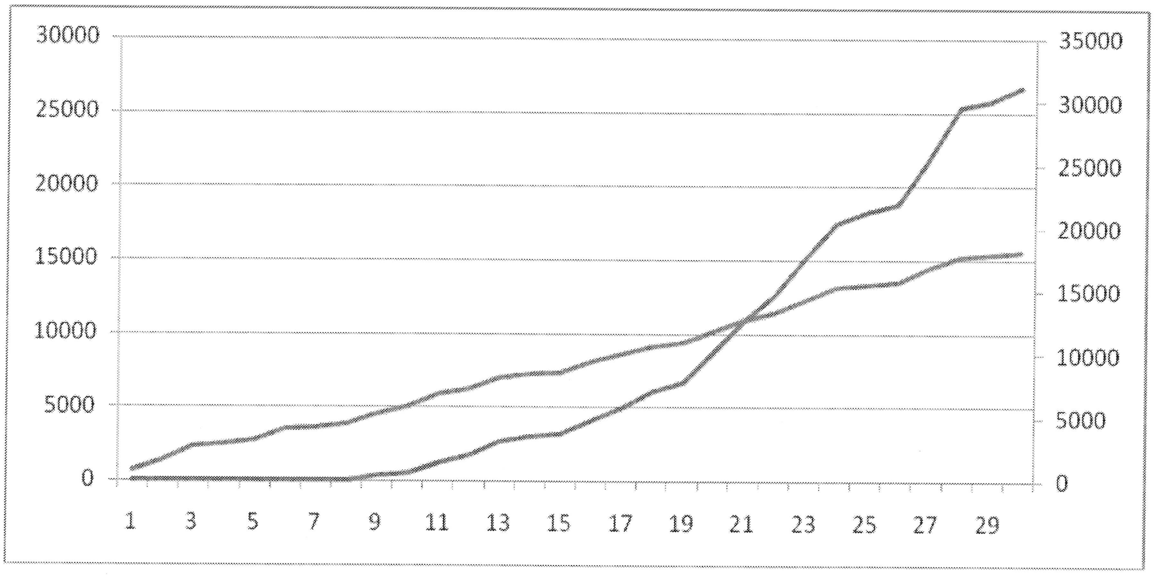
(Question 1 - continued on next page)

Power planners operate their stations in the following manner: Power stations are listed in order of least damage. (Where two stations do damage at the same level the order in which they are listed doesn't matter.) Depending on the power required at any time, stations are turned on in the order in which they are listed. A station is only turned on when all the stations before it in the list are operating at capacity. So, for a power load of 500 MW, only station 1 would be turned on. To provide 1000 MW, station 1 would be operated at 700 MW (full power), and station 2 would be run at 300 MW.

(a) Your first step is to enter the data from the power station table into a spreadsheet.

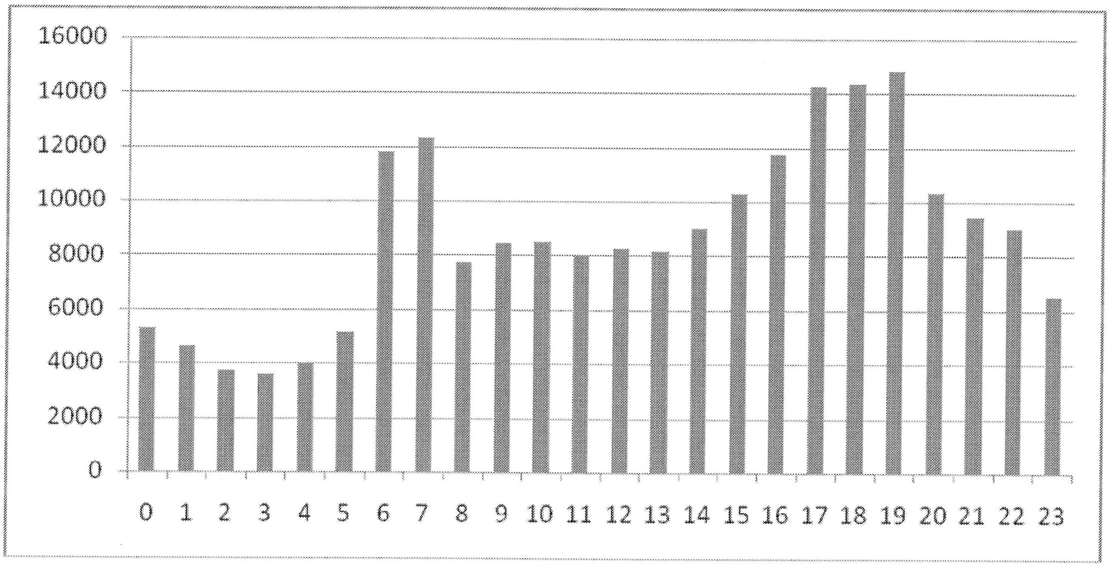
The first result you are asked to produce is a table, and graph (chart) that will give planners a rough idea of the output they can expect and the damage that will be caused by running stations at maximum power.

(b) Produce a table and graph showing the total maximum power production and the total environmental damage as a function of the number of power stations operating: ie: for 1 power station running the maximum power output would be 700 MW, with no damage; with two stations running the power would be 700 + 640 = 1340 MW, again with no damage. Your graph should look something like this (except that yours will be formatted and labelled nicely):

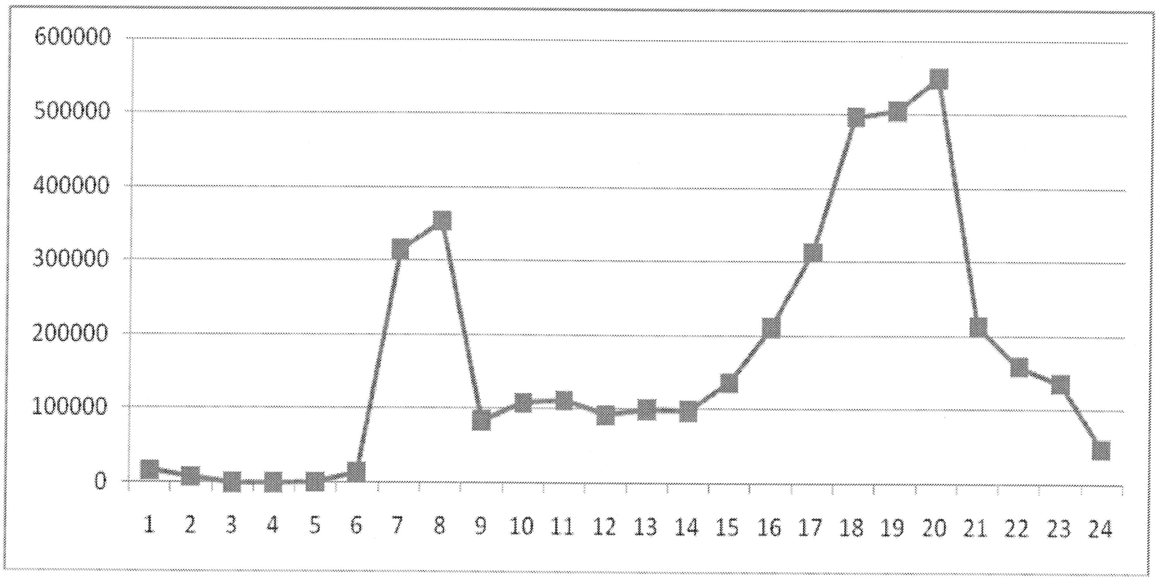


(Question 1 - continued on next page)

During the course of a typical day, power requirements in the country vary. They are usually at their lowest in the small hours of the morning and high during the day when businesses are active. Smaller peaks of usage occur in morning and evening. An example pattern of usage is given in the table on the next page, and looks like this (MW vs hour of the day) when graphed:



(c) Your third and more difficult task is to create a table and graph of the level of environmental damage being done during each hour of the day – for the example daily usage pattern given. Ie: at each hour you have to work out which power stations are running, and at what power levels. Use this information to calculate damage for each power station. Graph the total damage. It should look something like this:



(Question 1 - continued on next page)

Your spreadsheet should automatically update the graph if the order of power stations in the list is changed. The goal is to allow planners to experiment – on a year when water is scarce they might want to consider using hydro stations less, or they may want to know the cost of taking a power station out of service for maintenance, etc.

This is the table giving power usage for each hour of the day:

Hour	Power Requirement
0	5282
1	4630
2	3708
3	3593
4	3944
5	5145
6	11792
7	12332
8	7736
9	8429
10	8516
11	7994

Hour	Power Requirement
12	8238
13	8183
14	9021
15	10312
16	11759
17	14244
18	14343
19	14800
20	10330
21	9460
22	9019
23	6525

2. **Taxation (Careful and Accurate Programming)**

Your programming work in this question will be assessed on two criteria:

(a) Completeness and accuracy of the program.

(b) Good presentation. That is, it should make good use of programming language facilities, be well organised, neatly laid out, and lightly commented.

Your problem is to write a program to help a person work out their taxation situation as they are preparing their annual income tax return. The person has income from a number of jobs. From each job they know how much they earned during the year, and how much tax was taken from that income. They need to add up the income and tax values, work out the total tax actually owed on their total income and decide how much extra tax they need to pay, or if they have already paid more than enough, how much they can ask to get back.

The total tax owed by a person is calculated from their total income as follows: income up to $9,500 is taxed at 15%; income over $9,500 and up to a total income of $38,000 is taxed at 21%; income over $38,000 up to $60,000 is taxed at 33% and any income above $60,000 is taxed at 39%.

Examples:

1. An income of $5,000 is taxed at 15%, giving tax of $750

2 An income of $65,000 is taxed at;

	15% for the first $9,500 giving	$1,425
plus	21% for $9,500 to $38,000 giving	$5,985
plus	33% for $38,000 to $60,000 giving	$7,260
plus	39% for the last $5,000 giving	$1,950

for a total tax of $16,620

If the final tax amount includes fractions of a cent it is rounded to the nearest cent (0.5 cents is rounded up to 1 cent).

The input for your program consists of the number of jobs, and for each job the income and the amount of tax paid. For example:

```
4
10000.50 3020.28
20222.40 6500.00
15782.88 2000.27
19000.51 5000.81
```

The output from your program should be something like the following. The last line should use the text "Additional tax to pay", "Tax rebate due" or "Tax paid was correct" as appropriate. For example:

```
Total income = $65006.29
Total tax paid = $16521.36
Total tax due = $16622.45
Additional tax to pay = $101.09.
```

3. **Timetable (Problem Solving and Programming)**

Your programming work in this question will be assessed on two criteria:

(a) Your approach to the problem. We will be looking at your work for evidence that you found good ways of storing the necessary data, and devised algorithms for finding and displaying the requested results. Please hand in any notes and diagrams which describe what you are attempting to program, even if you don't have time to code or complete it.

(b) The extent to which your program works and correctly solves the problem.

Note also that this question asks for written answers to two questions. These are important because the problem being solved is too complex for you to produce a complete solution in the time available. In addition to seeing how far you can get with the problem, we want to see evidence that you have ideas for a more complete solution.

Please read through the whole question before starting work. There are suggestions as to how you should proceed at the end.

Computers are often used to make up timetables for examinations or classes. It is straightforward to programme a computer to check that a timetable is free of clashes. In general it is quite difficult to actually make a timetable because there are usually a large number of constraints to take into account. In a school timetable, for example, it is not just a matter of finding any way to fit students, teachers and classes into times. All sorts of extra considerations apply. For example, there may be a part time teacher who needs to have all their teaching on one day of the week. Often it is not possible to make a timetable completely without clashes. It may be necessary to settle for a timetable with the smallest possible number of problems. A good timetable program will select the best timetable it can find, while satisfying a range of conflicting requirements.

In this question you are asked to write a program to produce an examination timetable. You will be given a list of courses to be examined (identified by number); a list of students (also identified by number); and for each student, a list of the courses in which they are enrolled. Your task is to come up with an examination timetable. Your timetable will begin on Monday morning. Two examination time-slots are available each day - morning and afternoon. There is no restriction on the number of examinations that can be run in the same time-slot other than that a student can only be at one examination at a time. There is no restriction on the number of days you can use, but your goal is to find a solution which uses the least number of days.

For example. Consider a problem with 3 classes: 101, 102, 201 and 3 students: John, Mary and Peter, where Peter is enrolled in 101 and 102; Mary is enrolled in 201 and John is enrolled in 101 and 201. A possible timetable is:

Monday morning: 101: Peter, John
Monday afternoon: 102: Peter
Tuesday morning: 201: Mary, John

Notice though, that 102 and 201 could be scheduled at the same time as there is no student enrolled in both courses. So another possible timetable is:

Monday morning: 101: Peter, John
Monday afternoon: 102: Peter
201: Mary, John

The second timetable is more satisfactory because it fits into one day.

(Question 3 - continued on next page)

We have supplied you with a data file called prac3data.txt holding a sample timetable problem. It has data for 24 classes. Classes are identified by numbers from 100 to 123. For each class there is a list of the students enrolled. Students are identified by numbers starting at 1000 (just because it is easier to program with numbers than with names). The file looks like this:

```
24
100 25
1002 1003 1021 1027 1030 1033 1039 1047 . . . 1138 1152
101 27
1002 1003 1021 1027 1030 1033 1041 1047 . . . 1148 1152
. . .
. . .
123 32
1000 1004 1009 1017 . . . 1153
```

The first line is the number of classes. For each class there is a line with the class number, and the number of students in the class, then a line with a list of the students enrolled.

Tasks:

(a) Build a programme to solve the timetable problem. We suggest you proceed as follows

- Write instructions to read and store the class data.
- Write instructions to display your class data on the screen – to verify that you have read it correctly.
- Write instructions to make a clash array – ie: an array which shows you whether any two classes 'clash', ie: cannot be timetabled at the same time. Two classes clash if at least one student is in both classes.
- Write instructions to display your clash information.
- Devise an algorithm to form a timetable, using the clash information.

(b) Provide a written answer to the following question: "How would your algorithm scale: ie: how much longer would it take to run if you increased either the number of classes or the number of students? Would you expect it to be feasible for a university with 2,000 classes and 10,000 students."

(c) Provide a written answer to the following question: "For a complete timetable program, particularly one producing a timetable that might extend over a period of two or three weeks, what additional constraints should you take into account, and how do you think they could be specified in the programme's user interface?"