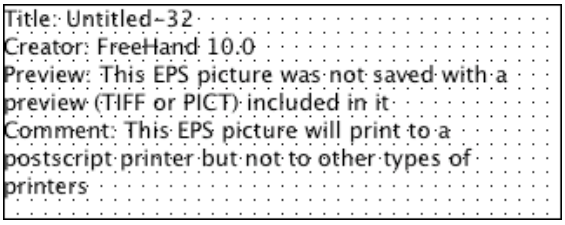


2006 SCHOLARSHIP EXAMINATION

PRACTICAL SECTION

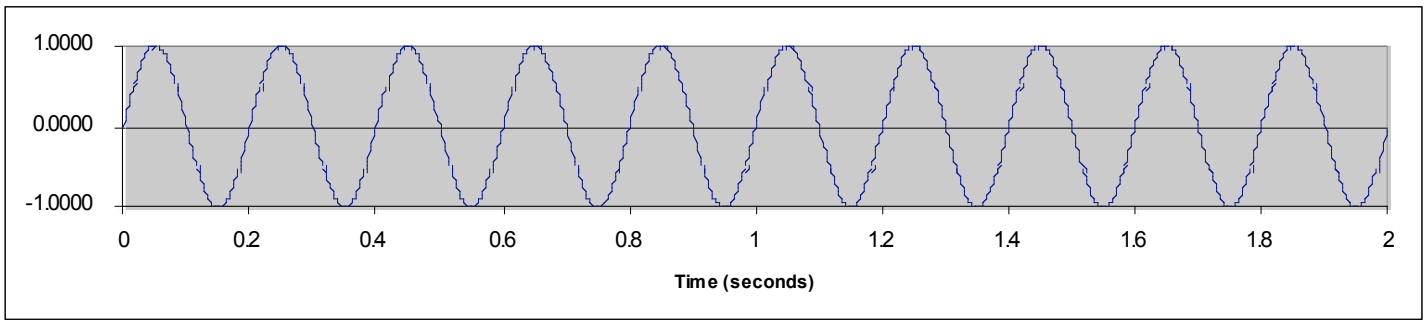
DEPARTMENT	Computer Science
PAPER TITLE	7th Form Scholarship
TIME ALLOWED	Six hours with a break for lunch at the discretion of the Supervisor.
NUMBER OF QUESTIONS IN PAPER	Three
NUMBER OF QUESTIONS TO BE ANSWERED	Three
VALUE OF EACH QUESTION	All question are of equal value.
GENERAL INSTRUCTIONS	Candidates 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 THREE is the most difficult and will take considerably longer than the others.
SPECIAL INSTRUCTIONS	Please hand in listings and notes for each question, and a floppy disk containing your program/computer work. 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.

QUESTION 1: SPREADSHEET USE – A NOTCH FILTER

This question will be marked on two criteria: The first is doing the required calculations and displaying the results clearly. The second is in using the spreadsheet to solve the problem in as general a way as possible – so that small modifications could solve related problems. If a value or a calculation were to be changed, your graphs should automatically update to show the new results. It should be possible to change parameter values in one place and have all calculations depending on that value update. Wherever something can be calculated, make the spreadsheet do it, don't work it out yourself and type solution values into spreadsheet cells.

Spreadsheets are most commonly used for financial calculations, but they can be used for calculation in any domain. For example the initial calculations for the path to Saturn followed by NASA's Cassini spacecraft were done on a spreadsheet. The example in this question is a signal processing experiment. Imagine that you are writing software to process digital music. When we play music we often want to adjust the tone, to emphasise bass (low frequency notes) or treble (high frequency notes). In digital sound processing the algorithms we use for this (and other) processes are called 'filters'. A low pass filter removes treble (by letting mostly low frequency notes through); a high pass filter removes base. In this exercise you will experiment with a 'notch' filter, which can be set to remove a part of the sound at a chosen band of frequencies – either treble, bass or somewhere in-between.

If we draw a graph of signal intensity against time for the simplest possible audio signal (a single tone, like the sound made by a tuning fork) it looks something like this:



Mathematically this is a 'sin' curve. Signal intensity at time t is $\sin(2\pi ft)$, where f is the signal frequency measured in cycles per second. In the graph above there are 10 cycles (10 peaks and 10 troughs) in 2 seconds, giving a frequency of 5 cycles per second.

Note: Real music works with signals of higher frequency. We will experiment with low frequency signals so that our spreadsheets don't become impossibly large.

Step 1. Simulating an audio signal

Make a column of numbers 0, 0.002, 0.004, 0.006, etc all the way to 2.000 (There will be over 1000 rows in your spreadsheet. Modern computers are fast and have a lot of memory – don't be afraid to use it.) This column represents time, in 1000 steps from 0 to 2 seconds.

(Question 1 – continued next page)

Make a column beside your time column with values of $\sin(2\pi ft)$ using the value 5 for the frequency *f*. This column is the signal intensity column ('signal' column for short).

Note: If you read ahead you will find that we need to experiment with different values of f later, so make sure it is easy to change f.

Make a graph of signal against time. You should get something like that shown above. Try different frequencies (*f*). Set the frequency *f* back to 5 for the next step.

Step 2. Parameters for the Notch Filter

People using the digital Notch Filter control it with two parameters. One is called *c*, the 'centre frequency'. This is the signal frequency most affected by the filter. The other is called *q*, and controls the width of the group of frequencies affected by the filter. In our experiments *c* will take values in the range 1 to 30. Values for *q* always lie between 0 and 1. To start with, you should try the value 20 for *c* and 0.9 for *q*.

The Notch Filter calculation uses six values that are calculated from *c* and *f*. They are:

$$z = \cos(2\pi c / 500)$$
$$Left_1 = q + (1-q)^2 / (2|z| + 1)$$
$$Left_2 = -2z * Left_1$$
$Left_3$ is the same as $Left_1$
$$Back_1 = 2zq$$
$$Back_2 = -q^2$$

Add to your spreadsheet as necessary to perform these calculations. You can use the value 3.14159 for π. $|z|$ means the absolute value of *z*, for example $|-3| = 3$, $|3| = 3$.

The notch filter works as a kind of rolling average. The process is shown in the diagram following. In my spreadsheet the new column begins at F18 (your layout may be different). We will write cell F18 as F_{18} because it makes the long expressions easier to read.

We begin by putting two zero values at the top of the new column (in cells F_{16} and F_{17}).

The first proper value in the new column is at F_{18}. It is computed from the three values to its left and lower left (E_{18}, E_{19} and E_{20}); and the two values above it (F_{16} and F_{17}) using the expression:

$$Left_1 * E_{18} + Left_2 * E_{19} + Left_3 * E_{20} + Back_2 * F_{16} + Back_1 * F_{17}$$

where $Left_1$, $Left_2$, $Left_3$, $Back_2$ and $Back_1$ are as calculated in Step 2.

(Question 1 – continued next page)

	D	E	F
14	Time	Signal	
15			
16			0.0000
17			0.0000
18	0.0000	0.0000	0.0033
19	0.0020	0.0628	
20	0.0040	0.1253	
21	0.0060	0.1874	
22	0.0080	0.2487	

Note: Sample data shown here is based on values: $c = 20,\ f = 5$ and $q = 0.9$

The next value is at F_{19} and it follows the same pattern, being computed from (E_{19}, E_{20}, E_{21}, F_{17} and F_{18})

$$Left_1 * E_{19} + Left_2 * E_{20} + Left_3 * E_{21} + Back_2 * F_{17} + Back_1 * F_{18}$$

	D	E	F
14	Time	Signal	
15			
16			0.0000
17			0.0000
18	0.0000	0.0000	0.0033
19	0.0020	0.0628	0.0125
20	0.0040	0.1253	
21	0.0060	0.1874	
22	0.0080	0.2487	

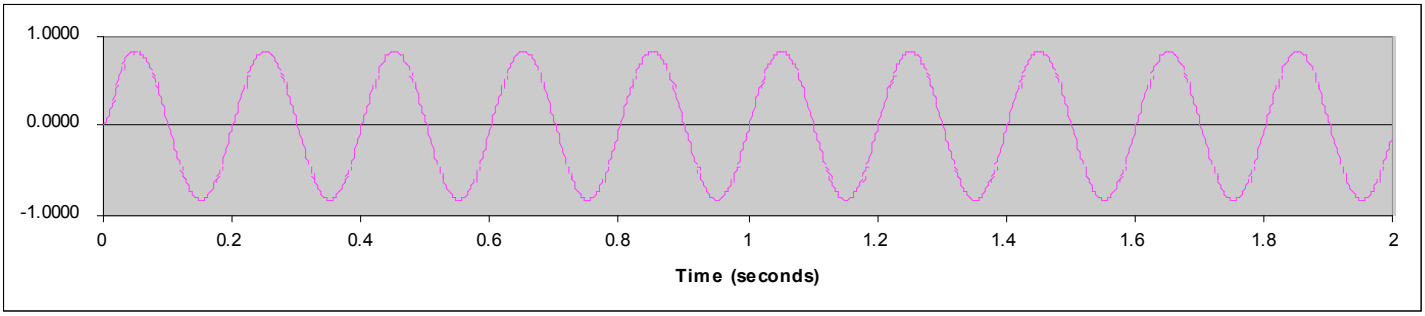
and so on to complete the column.

(Question 1 – continued next page)

Step 3. Implementing the Notch Filter

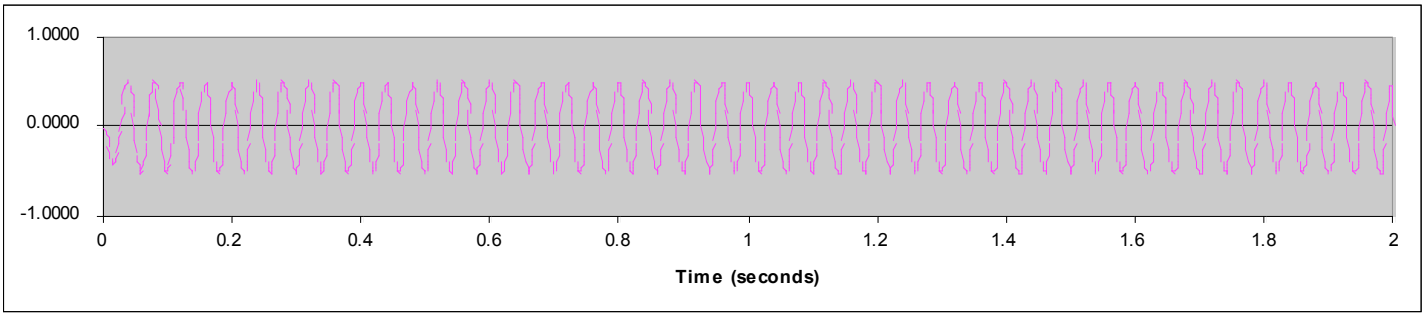
In a new column to the right of your signal column calculate a filtered signal as described above. Modify your graph to show both the signal and the filtered signal.

Try changing the frequency f of your signal. You should find that for values of f near c the filtered signal is reduced in intensity, and that for values of f that are not near c, the filtered signal is not much different from the original signal. For $f = 5$, $c = 20$, and $q = 0.9$ the filtered signal looks like this:



Step 4. Results

Your task in this section is to show how the intensity of the filtered signal varies for given values of q and f. For example, with $f = 25$, $c = 20$, and $q = 0.9$ the filtered signal is:



Your programming work in both Questions 2 and 3 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.

If you are uncertain about a particular detail in either of these questions write down the assumptions you are making. The questions will be marked given your assumptions should there be a genuine ambiguity.

QUESTION 2: PROGRAMMING – DIVISION

Write a program to read some numbers as illustrated in the sample interaction at the bottom of this page.

The program should read numbers until a negative value is read – the negative number signals the end of input, and is not part of the data to be processed. You may assume that no more than 100 numbers will be entered.

Think of the numbers as being written along a line, from left to right. (All the numbers in the line will be positive or zero.) Your task is to choose a dividing point to separate them into two groups, so that the sum of the numbers on the left is as nearly equal to the sum of the numbers on the right as possible. In cases where more than one answer is possible, choose the answer with the least number of values to the left.

For the example below we have

1 1 1 2 3

If the dividing point is put between the third 1 and the 2 we get

1 1 1 (sums to 3) | 2 3 (sums to 5)

If the dividing point is put between the 2 and the three we get

1 1 1 2 (sums to 5) | 3 (sums to 3)

which is no worse, so we choose the option with just the three 1's on the left.

Your program must decide on the position of the dividing line and display it as illustrated below.

Example interaction with the program

```
Please enter your numbers
1 1 1
2 3 -1
5 numbers were entered
Keep 3 numbers on the left hand side
```

QUESTION 3: PROBLEM SOLVING – PAGE RANKING

The value of the Internet depends almost entirely on the search engines provided by companies like Google and AltaVista. A search engine maintains a huge database of information about web pages. When we enter some search request the engine looks for pages that match the request. Often the number of matches is large, and the engine must decide which ones are the most likely to be useful to us. This decision process depends on being able to 'rank' or sort pages – to decide which pages provide the best answers to a search.

In this question you must find a solution to the page rank problem in a limited setting. Instead of complete web pages (with links and pictures, etc) our search engine will just hold a number of English sentences. We will assume that these sentences are all in lower case letters, have no punctuation, and are never more than 70 characters long. You can also assume that we will never have to deal with more than 100 sentences.

Example: Imagine that we hold the following 10 sentences ('web pages') in our program.

1. apples always have black pips in their cores
2. the apple computer company makes portable music players
3. an apple a day keeps the doctor away
4. apples are good for your health because they provide fibre
5. apple apple apple apple i just cannot get enough apples
6. i must demand that you do not slap please
7. pears and apricots are common at this time of year
8. frogs lay eggs in the water
9. toads are often brown and can live away from water
10. the apple toad has very long legs

Your system will accept search requests. Each request will be from 1 to 5 words separated by spaces (also in lower case, with no word longer than 14 characters).

eg: For the query "apple music", sentence 2 looks like the best response.

Develop your program in two steps

Step 1. Write a program

Write a program that:

(a) Reads and stores a number of sentences as described above.

(b) Reads queries of between 1 and 5 words.

(c) For each query, lists ALL sentences that include one or more words from the query, with the words of the query written in upper case letters. For "apple music" applied to the 10 sentences, this would produce:

the APPLE computer company makes portable MUSIC players
an APPLE a day keeps the doctor away
APPLE APPLE APPLE APPLE i just cannot get enough apples
the APPLE toad has very long legs

(Question 3 – continued next page)

Step 2. Devise a method of ranking search responses

Write a description of your ranking method. Note that this description may include features that you do not have time to implement, but you should describe your ideas anyway. Feel free to modify the solution that you programmed in Step 1 to include more of the sentences in your ranking process.

Implement your page ranking algorithm. It should display its results in the same style as in Step 1(c) above.