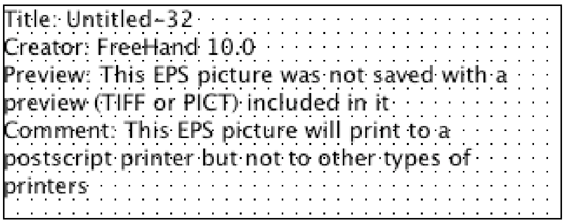


Title: Untitled-32
Creator: FreeHand 10.0
Preview: This EPS picture was not saved with a preview (TIFF or PICT) included in it
Comment: This EPS picture will print to a postscript printer but not to other types of printers

2004 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, notes and answers to written questions, and a floppy disk with your program / computer work for each questions. Please make sure that copies of programs are stored as plain text files. You can not 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.

TURN OVER

TURN OVER

1. **Power to Burn (Spreadsheet Use)**

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 - you will be marked down for performing calculations by hand and directly entering the results. Your work will be graded on three criteria.

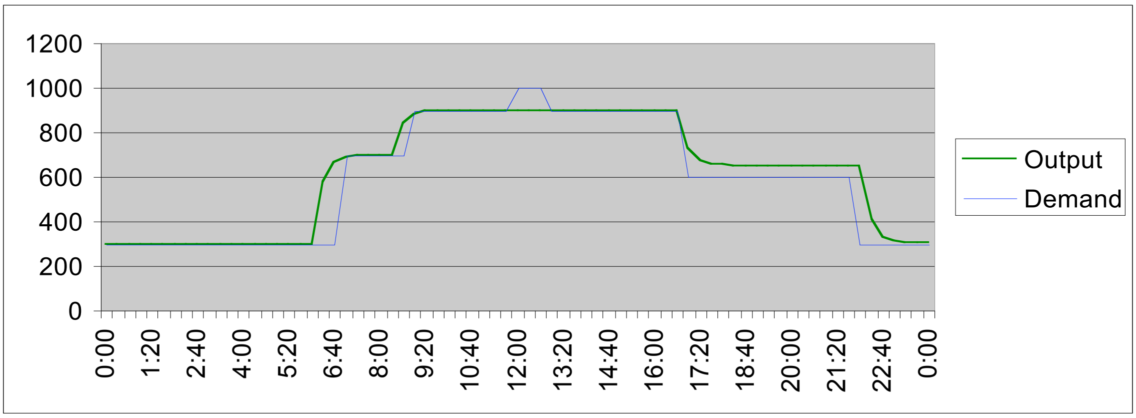
(a) The accuracy of your results.

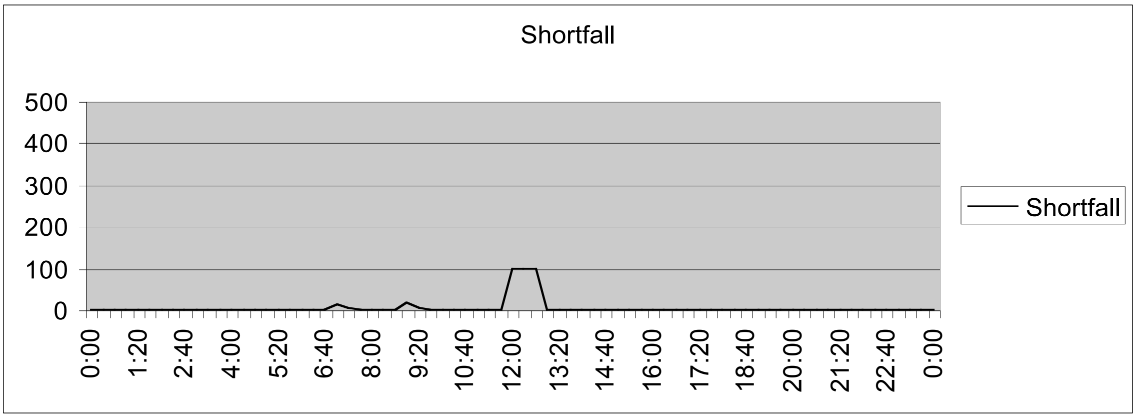
(b) The skill you show in making use of the capabilities of the spreadsheet.

(c) The presentation of your results. We have deliberately not provided any instructions concerning layout or formatting.

Spreadsheets are most commonly used for financial calculations. However they can be used for many other purposes. For example, the calculations needed to plan the current Cassini space mission to Saturn were done on a spreadsheet. A particular strength of spreadsheets is in doing 'what if' calculations – simulating some situation and showing 'what' happens 'if' some input is changed. In this question you are asked to develop such a spreadsheet to help train power station operators – showing them 'what' happens 'if' they set particular control values.

Operating a power station can be a difficult task, balancing demand for power with output of the station. One particular cause of difficulty is that changes in power output can lag quite some time behind changes in control values. The operator must therefore anticipate changes in demand and adjust controls so that power will be available when needed.





(Question 1 Continued On Next Page)

Specifically, you are to develop a spreadsheet to show power station operation over a period of 24 hours (from midnight one day to midnight the next). Proceed as follows.

- A perfect simulation would measure change continuously. For this exercise we will calculate average values over 20 minute periods. Therefore your spreadsheet should have a row every 20 minutes: i.e. 0:00, 0:20, 0:40, 1:00, 1:20, etc.
- The pattern of demand for power during the day is known. Add a column to show power demand.
  - From midnight to 6:40 the demand is 300MW
  - From 7:00 to 8:40 it is 700MW
  - From 9:00 to 11:40 it is 900MW
  - From 12:00 to 12:40 it is 1000MW
  - From 13:00 to 16:40 it is 900MW
  - From 17:00 to 21:40 it is 600MW
  - From 22:00 to midnight is 300MW
- There is only one control – it sets the fuel input rate for the station and can be set to any value from 0 to 100 inclusive. Arrange a column of your spreadsheet in which the trainee operator can enter input values. Once an input value is set, it should apply until a time at which another change is made. A challenge here is to make input setting easy for the user.
- The power station can deliver up to 1000 MW. After an input change the station output will come to 10 * the input value (it may take a while to reach this level). For input of 10, output will eventually reach 100, etc. Full output of 1000 MW results from input 100.
- The power station is slow to change its output level after a change in input. The average output for any 20-minute period will have shifted 70% of the way from the value it had at the last interval, to the value appropriate for the current input setting. For example: with the station operating at input 30 and output 300MW, setting the input to 40 would shift the output to

  300 + 70% of (400 – 300) = 370MW for that interval

  and if the input stayed at 40, output would shift to

  370 + 70% of (400 – 370) = 391MW in the following interval
- Initial operating conditions at midnight are: input 30, output 300MW.
- Add a column to show output
- Add a graph (chart) that shows demand and output.
- Add a column to show 'shortfall' – the amount by which output falls short of demand. This column should have the value zero when output is greater than or equal to demand.
- Add a column to show 'waste' – the amount by which output exceeds demand. This should be zero when there is no waste.
- Add charts to show shortfall and waste.

TURN OVER

2. Smoothing (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.

Sometimes computers are used to process information from unreliable sources. For example, imagine a computer system using measurements of boat position obtained from a sonar system to update a graphic display of the state of the race. Most of distance measurements coming from the sonar are accurate, but occasionally something will go wrong: a sonar signal might bounce off a passing seagull, giving a result that is too low; or a sonar pulse could be missed by the electronics, giving too large a distance. The graphic display would be unsatisfactory if boat positions jittered about. The answer is to 'smooth' the incoming data. Many approaches are possible. In this question you are asked to implement a data-smoothing algorithm (which is described below).

Your task is to write a program to read a series of numbers (integers in the range 0 .. 999) from the keyboard, and display a 'smoothed' version of the series.

The smoothing algorithm is as follows.

Each value is replaced by the median of itself, the value before it and the value after it.

For example in the series 2 3 9 4 5:

> The 3 remains unchanged because it is the median (middle value) of 2, 3 and 9;
>
> The 9 gets changed to 4 because 4 is the median of 3, 9 and 4;
>
> The 4 gets changed to 5, which is the median of 9, 4 and 5.

The first and last numbers in a series (2 and 5 in the example above) need to be treated specially. The first value should be replaced by the median of the first three numbers (3 in the example). The last value should be replaced by the median of the last three numbers (5 in the example).

Input and output for the program should work as illustrated on the next page. Values entered by the program's user are underlined. You can assume that the number of values will always be greater than 3 and less than 100.

(Question 2 Continued On Next Page)

```
Enter number of values in series: 15

Enter value #1: 21
Enter value #2: 22
Enter value #3: 23
Enter value #4: 3
Enter value #5: 25
Enter value #6: 26
Enter value #7: 27
Enter value #8: 29
Enter value #9: 31
Enter value #10: 32
Enter value #11: 30
Enter value #12: 28
Enter value #13: 87
Enter value #14: 23
Enter value #15: 21

Results for 15 values

     #   Original  Smoothed
     1       21       22
     2       22       22
     3       23       22
     4        3       23
     5       25       25
     6       26       26
     7       27       27
     8       29       29
     9       31       31
    10       32       31
    11       30       30
    12       28       30
    13       87       28
    14       23       23
    15       21       23
```

3. **Cracked Paragraphs (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.

Word processors have to be carefully programmed to layout text neatly. One of the most important algorithms is the 'justify' (or fill) algorithm. Your task is to develop a justification algorithm and then to investigate some of its more subtle details.

Text Justification: Adding extra spaces between words to make each line the same length.

For example, part of a paragraph taken from a NASA press release looks like this, without justification (called ragged-right).

```
Scientists are not sure if the object is alone. This is
because of results from a search through other images that
might capture the object to pin down its orbit. The search by
Dr. Joseph  Spitale, a planetary scientist working with team
…
```

With justification this becomes :

```
Scientists  are  not  sure  if  the  object  is alone. This is
because  of  results  from  a search through other images that
might  capture the object to pin down its orbit. The search by
Dr.  Joseph  Spitale,  a planetary scientist working with team
…
```

Note: Your justification algorithm will assume that all letters are the same width and will add extra whole spaces. A font with all characters the same width is called a 'non-proportional' or 'fixed-width' font. Such fonts are the norm in command interfaces – so if you write a program that reads and writes on a command line interface, that is what you should have. If your system uses a proportional font, just ignore it. The output won't look right, but you should be able to get a program to implement the algorithm. If you are finding it difficult to visually inspect the output, you could write a function that displays the number of characters in each line. Please mention any difficulties of this nature in your answer book and we will take the problem into account when marking.

Please develop your program in two stages.

(Question 3 Continued On Next Page)

Stage One: Develop a program in the following steps:

- Write a program to read a paragraph of text. The text can be organized into lines in whatever way the person entering the data likes, with the proviso that words cannot be split across line boundaries. The end of the paragraph should be marked by a blank line. The total length of the paragraph may be no more than 2000 characters.

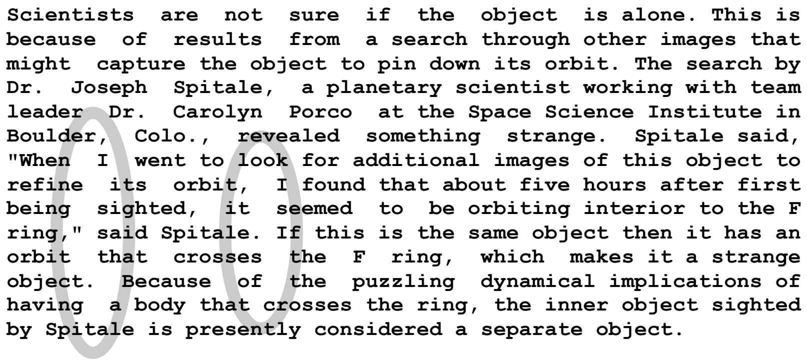
- Read a single number. This is the column width for justification. It should be in the range 30 to 80 (inclusive). Modify your program to break the input text into words and display the text in lines that have as many words as possible without exceeding the specified column width. Don't justify the text – just display as ragged-right. Words are made up of any character other than space or end-of-line. No special action to deal with punctuation is needed.

- Implement justification and test with different column widths. Use the following justification algorithm:

- If a line is too short by one character, add an extra space between the first and second words. If it is too short by two characters, add extra spaces between the first and second, and second and third words. Proceed in this way until the line is the correct length. If you have added spaces between all words on the line and it is still too short start adding a second space between the first and second words, then between second and third, *etc*.

Stage Two: The open-ended part of the question.

The justification algorithm implemented in stage one has some flaws. Consider the following paragraph, justified with this algorithm to a width of 62 characters. The text is nice and dense on the right side of the paragraph. On the left it looks 'loose' and there are unsightly 'cracks' where the extra spaces have been put (nearly) one above the other.



```
Scientists  are  not  sure  if  the  object  is  alone. This is
because  of  results  from  a search through other images that
might  capture the object to pin down its orbit. The search by
Dr.  Joseph  Spitale,  a planetary scientist working with team
leader  Dr.  Carolyn  Porco  at the Space Science Institute in
Boulder,  Colo.,  revealed  something  strange.  Spitale said,
"When  I  went to look for additional images of this object to
refine  its  orbit,  I found that about five hours after first
being  sighted,  it  seemed  to  be orbiting interior to the F
ring," said Spitale. If this is the same object then it has an
orbit  that  crosses  the  F  ring,  which  makes  it a strange
object.  Because  of  the  puzzling  dynamical implications of
having  a body that crosses the ring, the inner object sighted
by Spitale is presently considered a separate object.
```

Justification is the sort of problem that people are good at, and can solve quite easily, but that is difficult to program a computer to solve. We want you to think about the flaws in the algorithm coded in stage one, propose a solution, or perhaps several solutions, and then see what you can achieve in implementing your idea(s). A full answer will consist of brief written answer discussing your ideas about the problem, and as much of an implementation as you can achieve.

Special Note: Some computer systems may have library routines to perform text justification. It is not acceptable to use such routines. You are required to program the line justification yourself.