PRIFYSGOL CYMRU; UNIVERSITY OF WALES

M.Sc. & DIPLOMA EXAMINATIONS SUMMER 2002

SWANSEA

*Students may refer to any notes or books, but are **not** allowed to re-use any existing files stored on floppy disks or on the network.*

Computer Science

Module: **CS_M31**
PROGRAMMING

The examination is in two parts.

From ***10.00am-12 noon***, you should ***design*** your solution to the following problem. **Students are not permitted to leave this part of the examination early.** Your design should be handed in at the end of this time. It will be returned to you at the start of the second session ***(1.00pm-5.00pm)*** when you will have access to a PC. For this session, you will be supplied with a formatted disk on which there will be several files, the contents and purposes of which are detailed in the examination question. You will be expected to implement the solution that you designed in the first session, using the data on the provided disk, and to save your final solution on the disk for submission.

You should label this disk with your candidate number, your examination user name, and optionally your full name. You must save all of your code and your results files on to the disk provided. The version of your program you wish to be considered as your submission should be saved in to a file called **EXAM.DPR**. All other .DPR or .PAS files on the disk will be ignored (unless they are units utilised from within EXAM.DPR). This disk should be handed in at the end of the exam, along with your design document, plus any other comments you would like to make indicating problems, progress, changes in the design etc.

The question comprises **five** parts. You should aim to complete all parts in the time allotted. Your solution will be marked on style, correctness and completeness.

The exam question is not designed to be demanding, it is intended to give you an opportunity to demonstrate your understanding of what constitutes good style in programming. Do not waste time implementing code that is not required by the question. This includes the coding of elegant user interfaces. As a rule of thumb: Unless it is explicitly required by the question, leave it out.

Save your work regularly, no account can be taken of work that is lost due to a computer crash during the exam.

Time allowed: **2 hours** and **4 hours**

CS_M31
PROGRAMMING
(*Attempt **all** questions*)

Background

You have been contracted by the CIA to create a tool to help with deciphering substitution codes.

The encrypted sample material that you will be provided with contains spaces and uppercase letters only. The spaces can be ignored – as they have not been encrypted – and hence do not need to be decrypted.

You are required to write software that will analyse the letter frequency of a document, and suggest a letter for letter substitution that will decode the document. As it is unlikely that this will be correct on the first guess – you must provide the user with the facility to fine tune the substitution table manually in an iterative process.

The full process is described as follows:

1) Read the *English letter frequency table* from disk (provided in file *frequencytable.txt*)
2) Read the encoded document (from file *encoded.txt*)
3) Calculate the *letter frequency table* of the encoded document
4) Sort the document's *letter frequency table*
5) By inferring that high frequency letters in the encoded document probably match high frequency letters as given in the *English letter frequency table* produce an*automatically generated substitution table*
6) Display on screen the encoded document, using the *automatically generated substitution table* to decipher it
7) Give the user the opportunity to "tweak" the substitution table by reassigning pairs of enciphered letters
8) Repeat step 6 and 7 until document is completely decoded
9) Write to disk the *substitution table* (*subtable.txt*) and the decoded document (*decoded.txt*)

Example
An example using a simplified 5-letter alphabet is given below:

1) The *English language letter frequency table* is simply a list containing all of the letters of the alphabet in frequency order (determined by scanning a large number of English language documents). For example in our 5 letter alphabet:

    D {Most numerous character}
    E
    A
    B
    C {Least numerous character}

2) Our encrypted document is now scanned, and it is found to contain the following letters:

    T, G, F, R, M

    The Encrypted Document is:
    GTR RTFR GFRT GTFR RTTR RFG MFG

3) Counting the occurrences of these letters we get the following table:

Letter	*Frequency*
G	5
T	6
R	7
F	5
M	1

4) Sorting into descending order by number occurrences in the file we get our *letter frequency table* as shown below:

Letter	*Frequency*
R	7
T	6
G	5
F	5
M	1

5) Linking the two together gives us our *automatically generated substitution table*:

Original	*Cipher letter*
D	R
E	T
A	G

B	F
C	M

6) Using this table we get an initial decipher attempt:

GTR RTFR GFRT GTFR RTTR RFG MFG {Cipher text}
=
AED DEBD ABDE AEBD DEED DBA CBA {Deciphered text}

7) Assuming that all of the words in this document are real English words, then this is close, but not quite right. The word DEED looks OK, but the others all look a bit odd. Let us assume that we have correctly deciphered D and E because they look like they might be. We can then look at the first sequence of three letters which should be an English word. If we are right about the E and the D being correctly deciphered, then the only letter that can be correct in place of the A is a B (to make the word BED). So it looks as if G should be deciphered to B, not A.

So we manually change our *substitution table* so that the G is deciphered to B. This means that we have to associate A with a different letter. We always have to change two rows in the table, so we do a direct swap. (The A and the B change places)

The resulting table looks like this:

Original	*Cipher*
D	R
E	T
A	G
B	F
C	M

→

Original	*Cipher*
D	R
E	T
B	**G**
A	**F**
C	M

Our second decipher attempt is thus:

GTR RTFR GFRT GTFR RTTR RFG MFG {Cipher text}
=
BED DEAD BADE BEAD DEED DAB CAB {Deciphered text}

Suddenly it has all clicked in to place, and we have what is clearly a complete translation as all the words are valid words in English even if the sentence doesn't make any sense.

Questions – Attempt all Five

1) From the floppy disk provided, read in the *frequency table* (an ordered list of alphabetical characters) into an appropriate data structure. N.B: This list is provided, in ASCII text format on your floppy disk in the file **frequencytable.txt** .

[10 marks]

2) Read into an appropriate data structure the contents of the ASCII text file **encoded.txt**. From the text, create a *letter frequency table* using an appropriate data structure and sort the table by letter frequency.

[20 marks]

3) Automatically create a *substitution table* and decipher the encoded text using your generated *substitution table*, displaying the resulting decoded text on the screen for the user to look at. **[20 marks]**

4) Provide a facility to allow the user to display the current *substitution table*, and to swap letter pairs as required, generating a new *substitution table*. **[20 marks]**

5) Allow the user to apply the new table to the original text, displaying it on screen, and to repeat steps 4/5 until the translation is complete / intelligible. When this point has been reached, provide a facility to write to disk files:

a) The current *substitution table* (into a file **subtable.txt**)
b) The decoded text into a file (**decoded.txt**)

[30 marks]