Centre Number						Candidate Number				
Surname										
Other Names										
Candidate Signature										

General Certificate of Secondary Education
For submission in 2016

Computer Science 4512 — 4512/CB4

Unit 4512/1 – Practical Programming

Scenario 4: Traditional Application

Text Encryption

For candidates entering for the 2016 examination
To be issued to candidates on or after Tuesday 25 March 2014

*This scenario is one of four available. Each of the four scenarios is available in a separate candidate booklet. You must complete **two** of the four scenarios.*

- You have approximately 25 hours in which to complete this scenario.
- Before starting work on the problem, read the whole of this Candidate Booklet thoroughly. You can ask your teacher to explain anything in this booklet, except Computer Science specific terms, that you do not understand.
- There are restrictions on when and where you can work on this problem. Your teacher will explain them to you. For example, you should only do work that you intend to hand in for marking when a teacher is present, so that he or she can confirm that the work is your own. The Candidate Booklet must **not** be taken outside your school/college.
- You may need to use the Internet to research certain parts of the problem. This does not have to be within the 25 hours recommended time.
- You will need to complete and sign a Candidate Record Form which your teacher will provide.

Information

You will also be marked on your use of English. It is important to:

- make sure that all your work is legible
- use correct spelling, punctuation and grammar
- use a style of writing which suits the person you are writing for
- organise your information clearly, so that you make yourself understood
- use Computer Science terms where they are needed.

Scenario 4: Text Encryption

Encryption is the conversion of text into a form, called ciphertext, that cannot be easily understood by unauthorised people. Decryption is the process of converting encrypted text back into its original form, so it can be understood.

Encryption uses a key to hide the meaning of a message from people who are trying to read it. The encrypted message can only be understood if the key is used to change the encrypted message back to the original.

You have been asked to write a program that uses a randomly-generated key to encrypt the contents of a text file. The program should also allow the user to decrypt the contents of an encrypted text file if they provide the correct key.

The encryption method uses character substitution to encrypt the text. This is where one character is substituted for another based on an **offset factor**. For example if the **offset factor** is 3 that would mean that the character A would be substituted by the character D and so on.

Plaintext is the unencrypted text and ciphertext is the encrypted version. An example showing an extract from a character substitution table is shown in **Figure 1**.

Figure 1

Using an **offset factor** of 3

Plaintext	...	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	...
Ciphertext	...	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	...

Therefore using this example the words HELLO WORLD would be encrypted as KHOOR ZRUOG.

Tasks

1. Develop a main menu for the program. The options should be to:

   - encrypt message
   - decrypt message
   - exit the program.

2. Develop the part of the program that reads the contents of a text file to be **encrypted**.

   When the user selects the 'encrypt message' option from the menu created in **Task 1** the program should:

   - ask the user to enter the name of the file containing the message to be encrypted
   - load the contents of the specified file.

   You should use the **sample.txt** file provided by your teacher.

3. Develop the part of the program that generates an eight character key the program will use to encrypt the message.

   To generate an eight character key the program should:

   a. generate a random integer between 33 and 126
   b. convert the random number into its equivalent ASCII character (for example the number 65 would be converted into the character 'A')
   c. repeat steps a and b eight times to generate an eight character key
   d. display the eight character key to the user.

4. Calculate the **offset factor** based on the eight character key by converting each of the eight characters into its ASCII code, adding these together, dividing the result by 8, rounding down to a whole number and then subtracting 32. An example is shown in **Figure 2**.

Figure 2

Eight randomly generated numbers and their ASCII character.

Random Number	62	102	53	106	59	91	72	57
ACSII Character	>	f	5	j	;	[	H	9

When the eight random numbers are added together we get 602, then dividing by 8 gives 75.25. Rounding down gives 75 and then subtracting 32 results in an **offset factor** of 43.

5. Develop the part of the program that converts each character in the message into its ASCII code and encrypts it using the following rules:

   a. If the character is a space then do not encrypt the character
   b. otherwise:
      i. convert the character into its ASCII code (eg B = 66)
      ii. add the ASCII code to the **offset factor** calculated in **Task 4**. If the result is greater than 126 then subtract 94 so that the result is a valid ASCII code
      iii. convert the result into its equivalent ASCII character.

   An example is shown in **Figure 3**.

Figure 3

The plaintext letter B converts to the ASCII code 66, add the **offset factor** of 43 (from **Figure 2**) to get the result 109. As this is not more than 126 there is no need to subtract 94. The result, 109, is then converted into the ASCII character `m` (lowercase M).

   c. Create a string based on the results of a and b above. This is the ciphertext for the original plaintext message.

6. Develop the part of the program that saves the ciphertext into a new text file. The program should ask the user to enter the name of the new text file.

7. Develop the part of the program that reads the contents of a text file to be **decrypted**.

   When the user selects the 'decrypt message' option from the menu created in **Task 1** the program should:

   - ask the user to enter the name of the file containing the message to be decrypted
   - load the contents of the specified file
   - ask the user to enter the eight character key that was used to encrypt the message.

8. Develop the part of the program that decrypts the encrypted message using the eight character key entered by the user.

   a. Calculate the offset factor from the eight character key entered in **Task 7**.
   b. If the character is a space then do not decrypt the character
   c. Otherwise
      i. convert the character to its ASCII code
      ii. subtract the **offset factor** calculated in part a above from the ASCII code. If the result is less than 33 then add 94 so that the result is a valid ASCII code
      iii. convert the result into its equivalent ASCII character.
   d. Create a string based on the results of b and c above. This is the plaintext version of the original ciphertext message.

9. Develop the part of the program that displays the decrypted message to the user.

10. One of the problems with this type of encryption is that the word lengths for the encrypted message are the same as in the original message. In order to make cracking the encrypted message more difficult extend the program so that when encrypting the message the characters are grouped in blocks of five characters with a space separating each group, removing any spaces that were in the original message.

    This extended encryption should be provided as an additional option on the main menu. The extended encrypted message can be decrypted using the decryption method already developed. An example is shown in **Figure 4**.

Figure 4

Using an **offset factor** of 3

Plaintext	...	A	B	C	D	E	F	G	H	I	J	K	L
Ciphertext		D	E	F	G	H	I	J	K	L	M	N	O

M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	...
P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	

Therefore using this example the words `IT WAS A DARK AND STORMY NIGHT` would be encrypted as `LWZDU DGDUN DQGVW RUP\Q LJKW`

The decrypted message would be `ITWAS ADARK ANDST ORMYN IGHT`

In addition

1. Your Portfolio

You are free to use whatever software tools and techniques are available to you.

What your teacher will be looking for and how to provide that evidence for your Portfolio

In preparing you for this unit of work, your teacher will have provided you with more information about the section headings below.

Part 1 – Design of solution

Design of solution (0–9 marks available)
What you must do
• Show an understanding of what the problem involves with reference to the user's needs. • Produce an overview plan that shows how the problem is to be solved. • Produce pseudo code (or suitable alternative) showing the main blocks within the proposed solution.

Part 2 – Solution development

Solution development (0–9 marks available)
What you must do
• Show evidence of an understanding of how the final solution meets the needs of the user. • Produce annotated code that demonstrates an understanding of the programming techniques used.

Part 3 – Programming techniques used

Programming techniques used (0–36 marks)
What you must do
• Show an understanding of the programming techniques used and how the different parts of the solution work together. • Explain/justify the choice of programming techniques used to create a solution that has been coded efficiently. • Show evidence for the purpose and use of data structures. • Show the techniques used (appropriate to the language used) within the code to make the solution robust.

Part 4 – Testing and evaluation

Testing and evaluation (0–9 marks available)
What you must do
• Produce a test plan that shows the expected tests, test data and expected results. • Show that the planned tests have been carried out and provide a record of the actions taken. • Evaluate how the final solution meets the needs of the user.

2. Organising your Portfolio of work

Your Portfolio is where you keep the evidence that you have produced.

You should imagine that the Portfolio is to be used by another person who is interested in how you produced your solution. It is to help them to do something similar. It is important that you organise work for the Portfolio as shown below.

- You must keep all the work you produce for the organiser in hard copy in a Portfolio (or save your work electronically in folders which you will later copy onto a CD or DVD). Your teacher will have instructed you what to do.

- If you are putting hard copy printouts in your Portfolio make sure that you number each page and fasten it all together. Take your work out of any plastic sleeves before you hand it in to your teacher for marking.

- Each page should have your name, centre number and candidate number clearly shown on it.

- When you have completed this scenario, if you are putting your work on a CD or DVD, put the work for each heading on page **5** of this booklet in a separate folder. Each folder must be clearly named *(for example, 'Design of solution', 'Solution development' etc)*. Inside each folder the work must have a filename (*for example, 'What the problem involves', 'Control statements' etc*) which should be a final version for each heading. The CD or DVD should have your name, centre number and candidate number clearly shown on it. Your teacher will have advised you what to do.

END OF CANDIDATE BOOKLET