MSc (Computing Science) 2008-2009
C/C++ Laboratory Examination

Imperial College London

Tuesday 13 January 2009, 11h00 – 13h00

- ☞ You must complete and submit a working program by 13h00.
- ☞ Log into the Lexis exam system using your DoC login as both your login and as your password (**do not use your usual password**).
- ☞ You are required to write the header file **playfair.h**, and the corresponding implementation file **playfair.cpp**. You also need to create a **makefile** according to the specifications overleaf.
- ☞ Use the file **main.cpp** to test your functions. You will find this file in your Lexis home directory (**/exam**). If you are missing this file please alert one of the invigilators.
- ☞ **Save your work regularly.**
- ☞ The system will log you out automatically once the exam has finished. **It is therefore important that you save your work and quit your editor when you are told to stop writing.** No further action needs to be taken to submit your files – the final state of your Lexis home directory (**/exam**) will be your submission.
- ☞ No communication with any other student or with any other computer is permitted.
- ☞ You are not allowed to leave the lab during the first 30 minutes or the last 30 minutes.
- ☞ **This question paper consists of 5 pages.**

Problem Description

The *Playfair cipher* is an encryption system that was devised in the mid-19th century. A simplified version of this scheme works as follows:

1. A 6×6 encoding grid containing the letters of the alphabet A-Z and the digits 0-9 is set up. The order in which these 36 characters appear in the grid is determined by a codeword. The first occurrence of the letters or digits in the codeword appear first, followed by the unused letters and digits in lexical order. The left grid in Fig. 1 shows the encoding grid for the codeword `IMPERIAL`.

I	M	P	E	R	A
L	B	C	D	F	G
H	J	K	N	O	Q
S	T	U	V	W	X
Y	Z	0	1	2	3
4	5	6	7	8	9

I	M	P	E	R	A
L	B	C	D	F	G
H	J	K	N	O	Q
S	T	U	V	W	X
Y	Z	0	1	2	3
4	5	6	7	8	9

Figure 1: The grid for the codeword `IMPERIAL` (left), and how the bigram `CO` is coded as `FK` (right).

2. The message to be encrypted (the plain text) is then split into bigrams (groups of two letters). Spaces and punctuation in the plain text are ignored, and an X is added to the plain text to complete the last pair if necessary. Thus the message “`COME TO THE QUEEN’S TOWER AT 7 O’CLOCK.`” is divided up into the bigrams `CO ME TO TH EQ UE EN ST OW ER AT 7O CL OC KX`.

3. Each plain text bigram is then encoded into two cipher characters as follows. The first cipher character is the grid character at the intercept of the *row* of the first bigram character and the *column* of the second bigram character. The second cipher character is the grid character at the intercept of the *column* of the first bigram character and the *row* of the second bigram character. Thus the bigram `CO` is encoded as `FK` (as shown on the right in Fig. 1), and the resulting cipher text for the whole message is:

   `FKEMWJSJANVPENTSOWREMX8NLCKFQU`

Specific Tasks

1. Write a function `prepare(input,output)` which produces an output string suitable for Playfair encoding from some input string by:

   - copying the alphanumerical (i.e. letters and digits) characters (but not any space or punctuation characters) from the input string to the output string
   - making the letters in the output string all uppercase
   - adding an 'X' to the output string if it contains an odd number of characters

   The first parameter to the function (i.e. input) is a read-only string containing an input sentence. The second parameter (i.e. output) is the prepared output sentence.

   For example, the code:

   ```
   char prepared[100];
   prepare("Come to the Queen's Tower at 7 o'clock!",prepared);
   ```

   should result in the string `prepared` having the value
   `COMETOTHEQUEENSTOWERAT7OCLOCKX`

2. Write a function grid(`codeword, square`) which produces the 6x6 Playfair square (encoding grid) corresponding to a given code word. The first parameter to the function (i.e. codeword) is an input string containing a code word (e.g. `IMPERIAL`). You may assume that the code word consists of upper case letters and digits only. The second parameter (i.e. square) is an output parameter which takes the form of a two-dimensional array of characters representing the encoding grid.

   For example, the code:

   ```
   char playfair[6][6];
   grid("IMPERIAL",playfair);
   ```

   should result in the two-dimensional array `playfair` having the value shown on the left in Fig. 1.

3. Write a function `bigram(square, inchar1, inchar2, outchar1, outchar2)` which encodes a single bigram (two letter pair) using a given encoding grid. The parameters are as follows:

   - `square` is the encoding grid to be used to encode the bigram.
   - `inchar1` and `inchar2` are two character input parameters making up the bigram to be encoded.
   - `outchar1` and `outchar2` are two character output (reference) parameters representing the encoded bigram.

   For example, the code:

```
char playfair[6][6];
grid("IMPERIAL",playfair);
char out1, out2;
bigram(playfair,'C','O',out1,out2);
```

   should result in `out1` and `out2` having the values 'F' and 'K' respectively (as shown on the right in Fig. 1).

4. Write a function `encode(square, prepared, encoded)` which encodes a prepared input string using a given encoding grid. The parameters are as follows:

   - `square` is the encoding grid to be used.
   - `prepared` is the prepared input string containing an even number of upper case letters and/or digits.
   - `encoded` is an output parameter containing the encoded sentence.

   For example, the code:

```
char playfair[6][6];
grid("IMPERIAL",playfair);
char encoded[100];
encode(playfair,"COMETOTHEQUEENSTOWERAT7OCLOCKX",encoded);
```

   should result in the string `encoded` having the value:
   `FKEMWJSJANVPENTSOWREMX8NLCKFQU`

   For full credit, your solution should be recursive and use pointer arithmetic. However, partial credit (up to 75%) will be awarded for a working iterative solution.

Place your function implementations in the file **playfair.cpp** and corresponding function declarations in the file **playfair.h**. Use the file **main.cpp** to test your functions. Create a **makefile** which compiles your submission into an executable file called **playfair**.

(The four parts carry, respectively, 20%, 30%, 25% and 25% of the marks)

Bonus challenge

For a 5% bonus, write a function `decode(square,encoded,decoded)` which decodes a Playfair-encoded message.

Hints

1. Feel free to define any auxiliary functions which would help to make your code more elegant.

   In particular, when answering Question 2 you might find it useful to define an auxiliary function `bool occurs_before(const char str[], char letter, int pos)` which returns true if the character `letter` occurs in a string `str` before some position `pos`.

   Also, when answering Question 3 you might find it useful to define the functions `int find_row(const char square[6][6], char ch)` and `int find_col(const char square[6][6], char ch)` which respectively return the row and column of a particular character in an encoding grid.

2. The standard header `<cctype>` contains some library functions that you may find useful. In particular:

   - `int isalnum(char ch)` returns nonzero if `ch` is a letter from `'A'` to `'Z'`, a letter from `'a'` to `'z'` or a digit between `'0'` and `'9'`.
   - `char toupper(char ch)` returns the upper case equivalent of character `ch`.

3. Try to attempt all questions. Note that Questions 1 and 2 can be attempted independently, and if you have the function prototype for Question 2, you can write Questions 3 and 4.