MSc (Computing Science) 2005-2006
C/C++ Laboratory Examination

Imperial College London

Wednesday 11 January, 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 create a header file **multitap.h**, a corresponding implementation file **multitap.cpp** and a **makefile** according to the specifications overleaf.
- ☞ Use the file **main.cpp** and the sample input file **message.txt** to test your functions. You will find these files in your Lexis home directory (**/exam**). If you are missing these files 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

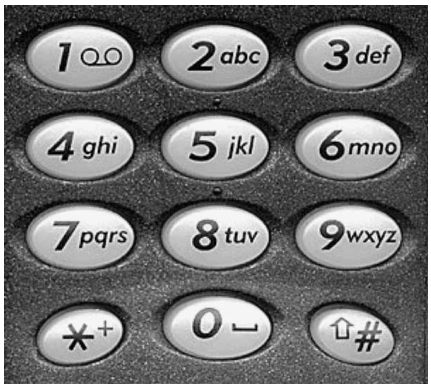

Figure 1: A mobile phone keypad.

Multitap is a method of text entry using a mobile phone keypad. As shown in Figure 1, a typical keypad consists of the number keys `0`–`9` and two additional keys (`*` and `#`). The letters `a`–`z` are spread over the keys `2`–`9` in alphabetic order, and the space character (‘ ’) is assigned to the `0` key.

With multitap, a user presses each key one or more times to specify a desired letter. For example, the `6` key is pressed once for the letter `m`, twice for the letter `n` and three times for the letter `o`. Thus the word `imperial` is encoded as `44467337774442555` in multitap.

Letter case is controlled using the `#` key, which toggles (i.e. switches) between upper case and lower case input modes; initially input begins in lower case mode. Thus `FOx` is encoded as `#333666#99`.

Note that, in order to avoid ambiguity, users may have to pause between consecutive letters (of the same case) that are encoded using the same key. For example, the word `cab` involves repeated presses of the `2` key. If we use a ‘`|`’ character to indicate a pause, then `cab` is encoded in multitap as `222|2|22`.

Punctuation marks are entered by repeated presses of the `1` key. In particular, you may assume that ‘`.`’ (full stop), ‘`,`’ (comma), ‘`!`’ (exclamation mark) and ‘`?`’ (question mark) are encoded on the `1` key (in that order).

Digits are entered by pressing the `*` key and then the corresponding number key. So `15` is encoded as `*1*5`.

Specific Tasks

1. Write a function `encode_character(ch,multitap)` which produces the multitap encoding of a single input character (ignoring case) and which also returns the number of keystrokes required to encode the input character. The first parameter to the function (i.e. `ch`) is the character to be encoded. The second parameter (i.e. `multitap`) is an output parameter that should contain the multitap-encoded string corresponding to the character. The return value of the function is the length of the multitap encoding.

   For example, the code:

   ```
   char multitap[20];
   int size;
   size = encode_character('c', multitap);
   ```

   should result in the string `multitap` having the value `"222"` and `size` having the value `3`.

   As another example, the code:

   ```
   char multitap[20];
   int size;
   size = encode_character('5', multitap);
   ```

   should result in the string `multitap` having the value `"*5"` and `size` having the value `2`.

2. Write a function `encode(plaintext,multitap)` which produces the multitap encoding of a plaintext input string. The first parameter to the function (i.e. `plaintext`) is an input string containing the string to be encoded. The second parameter (i.e. `multitap`) is an output parameter which should contain the corresponding multitap-encoded string (taking letter case and pauses into account).

   For example, the code:

   ```
   char multitap[100];
   encode("Meet Anna at 5pm", multitap);
   ```

   should result in the string `multitap` having the value:
   `"#6#33|3380#2#66|6620280*576"`.

3. Write a function `decode(input,output)` which takes a multitap-encoded input stream and writes a decoded plaintext version to an output stream. The first parameter (i.e. `input`) is the input stream (e.g. `cin` or a file input stream) and the second parameter is the output stream (e.g. `cout` or a file output stream).

   For example, given an input file called `message.txt` that contains the text:

   ```
   #222#666|66477728|8855528444666|667777111
   ```

   then the code:

   ```
   ifstream input;
   input.open("message.txt");
   decode(input, cout);
   input.close();
   ```

   should result in the following output written to `cout` (i.e. displayed on the screen):

   ```
   Congratulations!
   ```

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

(The three parts carry, respectively, 30%, 30% and 40% of the marks)

P.T.O. for hints

Hints

1. Feel free to define any auxiliary functions which would help to make your code more elegant. You might find this a particularly useful thing to do when answering Question 3.
2. The standard header `<cctype>` contains some library functions that you may find useful. In particular:
   - `int isalpha(char ch)` returns nonzero if `ch` is a letter from `'A'` to `'Z'` or a letter from `'a'` to `'z'`.
   - `int islower(char ch)` returns nonzero if `ch` is a letter from `'a'` to `'z'`.
   - `char tolower(char ch)` returns the lower case character corresponding to `ch`.
   - `int isdigit(char ch)` returns nonzero if `ch` is a digit between `'0'` and `'9'`.
3. Try to attempt all questions. If you cannot get one of the questions to work, try the next one.
4. You are not explicitly required to use recursion in your answers to any of the questions. Of course, however, you are free to make use of recursion if you wish (esp. where it increases the elegance of your solution).