DWITE Online Computer Programming Contest
November 2005

Problem 5

Base64 Encoding

Base64 encoding takes three bytes, each consisting of eight bits, and represents them as four printable characters in the ASCII standard. It does that in essentially two steps.

The first step is to convert three bytes to four numbers of six bits. Each character in the ASCII standard consists of seven bits. Base64 only uses 6 bits (corresponding to 2^6 = 64 characters) to ensure encoded data is printable and humanly readable. None of the special characters available in ASCII are used. The 64 characters (hence the name Base64) are 10 digits, 26 lowercase characters, 26 uppercase characters as well as '+' and '/'.

If, for example, the three character string "The" converts to the ASCII bytes 84, 104 and 101, the corresponding bit stream is 010101000110100001100101, which in turn corresponds to the 6-bit values 21, 6, 33 and 37.
These numbers are converted to ASCII characters in the second step using the Base64 encoding table.

Sequence	Characters
0 … 25	'A' … 'Z'
26 … 51	'a' … 'z'
52 … 61	'0' … '9'
62	'+'
63	'/'

The 6-bit values of our example translate to the ASCII sequence "VGhl".

- 84 -> 01010100
- 104 -> 01101000
- 101 -> 01100101
- 010101 -> 21
- 000110 -> 6
- 100001 -> 33
- 100101 -> 37
- 21 -> V
- 6 -> G
- 33 -> h
- 37 -> l

This two-step process is applied to the whole sequence of bytes that are encoded.

At the end of the encoding process we might run into a problem. If the size of the original data in bytes is a multiple of three, everything works fine. If it is not, we might end up with one or two 8-bit bytes. For proper encoding, we need exactly three bytes, however. The solution is to append enough bytes with a value of '0' to create a 3-byte group. Two such values are appended if we have one extra byte of data, one is appended for two extra bytes. Of course, these artificial trailing '0's cannot be encoded using the encoding table. They must be represented by a 65th character. The Base64 padding character is '='. Naturally, it can only ever appear at the end of encoded data.

The input file (**DATA51.txt** for the first submission and **DATA52.txt** for the second submission) will contain five lines of data. Each line will contain an Base64 encoded string. The string will not exceed 1000 characters in length.

The output file (**OUT51.txt** for the first submission and **OUT52.txt** for the second submission) will output for each input line the decoded original string.

Sample Input (Only three sets given)	**Sample Output**

`VGhl`
`VGhleQ==`
`SGVsbG8=`
	`The`
	`They`
	`Hello`

Analysis for second set of data:

Encoding: They
T - 01010100
h - 01101000
e - 01100101
y - 01111001
It is two bytes short so 0000000000000000 gets padded on the end.
The string gets put into groups of 6 bits
010101-000110-100001-100101-011110-010000-000000-000000
Which gets encoded as
VGhleQ== (groups of 6 bits that are all zeroes and appear at the end of the string encode as =)

Decoding: VGhleQ==
010101-000110-100001-100101-011110-010000-000000-000000
Put these into groups of eight bits
01010100-01101000-01100101-01111001-00000000-00000000
Ignore groups of eight bits that are all zeroes and at the end of the string
01010100-01101000-01100101-01111001
Find the character code for these binary ASCII values
They

http://email.about.com/cs/standards/a/base64_encoding.htm

http://makcoder.sourceforge.net/demo/base64.php