The 3rd Annual Newport News Computer Challenge

Thursday, February 19, 2004

The Base 64 Encoder (30 points)

E-mail protocol (SMTP, or Simple Mail Transfer Protocol) only allows 7 bits of information to be sent at a time. Of course, information must be transfered across the Internet in bytes. And bytes contain 8 bits. So in e-mail, the 8th bit is always 0. With 7 bits, we can only represent the numbers 0 (0000 0000 in binary) to 127 (0111 1111 in binary).

This is not a problem as long we only send text in our e-mail message, which was the original purpose. The 7-bit limit was also intended to provide security by preventing unintended executable files from being sent to an unsuspecting e-mail recipient.

Then along came the World Wide Web and desktop computers and more and more people began to use the computer for business and recreational purposes, and they wanted to attach 8-bit binary files (such as images, word processor documents, and even executable programs) to their 7-bit SMTP messages. But binary files contain bytes that use all 8 bits.

Enter, the problem.

How do you transfer up to 256 different bytes (0-255) with only 128 different bytes (0-127)? Well the easiest way would have been to simply split each byte in half and make two bytes. That way any byte, such as 1101 1001 (in binary), would end up being two bytes, 0000 1101 and 0000 1001, each of which would never be larger than 15 (0000 1111 in binary). To convert back, all we would need to do is multiply the first byte by 16 and add . But you know computer programmers. They just shuttered at the idea of doubling the size of every binary file attached to an SMTP message. They just hate to waste band-width. And besides, until recently transfer speeds using older modems was very slow.

Enter, Base 64.

Several different schemes for encoding binary into SMTP-friendly text were devised, but the one that seems to be most popular lately is called "Base 64". In this scheme, three bytes of binary information are converted into four bytes of text. Here's how:

First, there are 64 "base" characters (hence, the name "Base 64"). They are:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
```

The base characters are indexed beginning with 0 and ending with 63. So base character 0 is "A", base character 1 is "B", and base character 63 is "/", to name a few.

Suppose we have three bytes of information that look like this:
```
aaaa aaaa  bbbb bbbb  cccc cccc
```
The bits of the first byte are represented by 8 letter a's, the bits of the second byte are represented by 8 letter b's, and the bits of the third byte are represented by 8 letter c's.

To convert three bytes into four, we group the bits into groups of 6, like this:
```
aaaa aa|aa  bbbb| bbbb  cc|cc cccc
```
And then we place each group into a new byte whose first two bits are zero (0), like this:
```
00aa aaaa  00aa bbbb  00bb bbcc 00cc cccc
```
The result is four new bytes, each of whose smallest possible value is 0 (0000 0000 in binary) and whose largest possible value is 63 (0011 1111 in binary).

The last thing we do is look up the base character whose index is the value of each new byte. This results in four characters, each chosen from the list of base characters.

For example, suppose our three bytes are 83, 17, and 204. In binary, they would look like this:
```
0101 0011   0001 0001   1100 1100
```
If we group their bits into groups of six and put each group into a new byte whose first two bytes are 0, we get:
```
0001 0100   0011 0001   0000 0111   0000 1100
```
These respective values are 20, 49, 7, and 12.

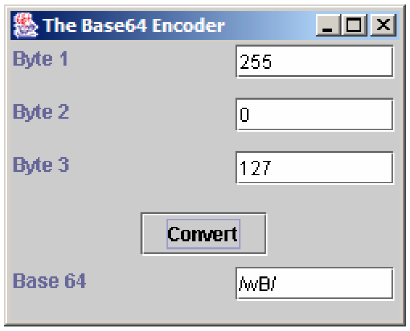
The four base characters located at these four indexes are "U", "x", "H", and "M".

Therefore, the Base 64 code for 83, 17, and 204 is "UxHM".

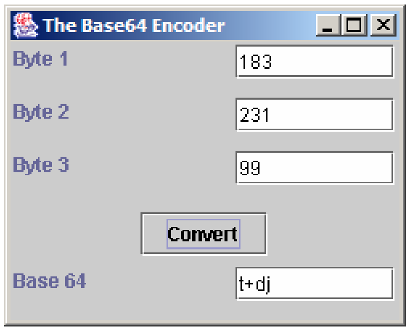
YOUR JOB

Your job is to write a Java GUI program that accepts three integers as input and outputs the resulting four characters of Base 64 code. The three integers must be within the range of 0 to 255 inclusive to be valid. If one or more is invalid, "INVALID" is displayed.

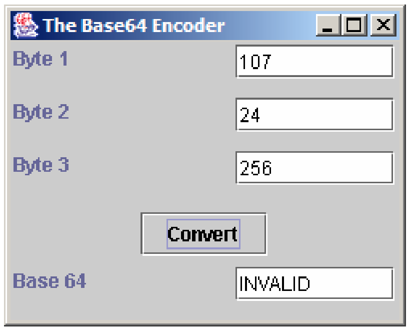
Samples:


The Base64 Encoder
Byte 1: 255
Byte 2: 0
Byte 3: 127

Base 64: /wB/


The Base64 Encoder
Byte 1: 183
Byte 2: 231
Byte 3: 99

Base 64: t+dj


The Base64 Encoder
Byte 1: 107
Byte 2: 24
Byte 3: 256

Base 64: INVALID