PROBLEM FIVE: ERROR CHECKING CODES

All Credit Card numbers follow certain patterns. A credit card must have between 13 and 16 digits and it must start with a 4, 5, 37, or 6.

In 1954, Hans Luhn of IBM proposed an algorithm for validating credit card numbers. This algorith is useful to determine if a card number is entered correctly or if a credit card is scanned correctly by a scanner. Almost all credit card numbers are generated following this validity check, which can be described as follows:

1. Double every second digit from right to left. If doubling of a digit results in a two-digit number, add up the two digits to get a single-digit number. For example, if a particular second digit were 4, then computing that digit would be $4 \times 2 = 8$. If a particular second digit were 6, then computing that digit would be $6 \times 2 = 12 \rightarrow 1 + 2 = 3$.

2. Now all all single-digit number from Step 1.

3. Add all digits in the odd places from right to left in the card number.

4. Sum the result from Step 2 and Step 3.

5. If the result from Step 4 **is** divisible by 10, the card number is valid. If the result from Step 4 **is not** divisible by 10, it is invalid.

Write a program that takes a credit card number as a long integer, entered as a command line arguement. Determine using Luhn's algorithm if the Credit Card number is valid or not.

Example valid Credit Card number: `4388576018410707`

Example invalid Credit Card number: `4388576018402625`

If you have an actually credit or a debit card, don't use the number for testing. We would like very much that your bank account remains secure, so please minimize the risk by keeping any actual credit or debit card numbers away from prying eyes. Use the example Credit Card numbers instead for testing.

REQUIRED INPUT: A Credit Card number between 13 and 16 digits, input as a command line argument.
REQUIRED OUTPUT: Compute the validity of the Credit Card number that was input using Luhn's algorithm and output the result of the validity test.