Assignment: 2

Due: Thursday, May 28 at 9:00 am
Language level: Beginning Student
Coverage: Modules 1–2

For this and all subsequent assignments, to receive full marks you are required to use the design recipe for every function you write. For your convenience, an interface file which contains the headers of the required functions is available on the course webpage.

Do not send any code files to course staff; they will not be accepted. Submissions must be made via MarkUs as described on the course webpage. After submission, check your basic test results to ensure your files were properly submitted. Solutions that do not pass the basic tests are unlikely to receive any correctness marks.

Remember, the solutions you submit must be **entirely your own work**.

1. Pig Latin is a simple way of disguising words. In this question, you will write a Racket function which performs a simplified Pig Latin translation. To translate a given word, move the first letter of the word from the start to the end, and append "ay" to the end of the word.

   Write a Racket function *pig-latin* which consumes a nonempty string *word* and produces a string containing the Pig Latin translation of *word*.

   For example, (*pig-latin* `"racket"`) should produce `"acketray"`.

2. To pass CS 115 you are required to pass the examination portion of the course; i.e., your weighted exam mark must be 50% or higher. Upon hearing this, you decide to write Racket functions which will make it easy for you to compute and understand this requirement in practice. (While simultaneously practicing what you've learned!)

   (a) Write a Racket function *weighted-exam-mark* which consumes two natural numbers between 0 and 100 (*midterm-mark* and *exam-mark*, in that order) and produces the weighted exam mark (as a percentage value) of *midterm-mark* and *exam-mark*. The midterm is worth 30% of the final grade and the final exam is worth 45%, so the weighted exam mark is

   $$\frac{30\% \times \textit{midterm-mark} + 45\% \times \textit{exam-mark}}{30\% + 45\%}.$$

   For example, (*weighted-exam-mark* `70` `80`) should produce `76`.

   (b) Write a Racket function *exam-mark* which consumes two natural numbers between 0 and 100 (*midterm-mark* and *weighted-exam-mark*, in that order) and produces the percentage value necessary on the final exam to achieve the requested weighted exam mark, given the knowledge of the midterm mark that was obtained.

For example, if you got 70% on the midterm and want a weighted exam mark of 80%, you must get at least 86.66…% on the final, so (*exam-mark* 70 80) should produce $86.\overline{6}$. Note the final exam mark may may be over 100% and therefore unattainable.

3. To guard against transmission and transcription errors, many types of account numbers contain a *check digit*, an extra digit in the account number which can be determined from the remaining digits. Before using an account number one should determine what the check digit should be if the number is accurate, and verify that this matches the actual check digit; if there is a mismatch then there must be a mistake in the account number.

   In this question, we'll write a Racket function *add-check-digit* which will consume an account number and produce the account number with a check digit included. The function will be written in stages; in each part you may rely on any function written in a previous part.

   (a) Write a Racket function *add-digit* which consumes a natural number n and a single digit d (i.e., $0 \leq d \leq 9$) and produces the number which contains all the digits of n, but with the digit d added on the far right.

   For example, (*add-digit* 1234 5) should produce 12345.

   (b) Write a Racket function *extract-digit* which consumes two natural numbers (n and k, in that order) and produces the kth digit of n. We'll consider the "ones digit" to be the 0th digit, and the "tens digit" to be the 1st digit, and so on.

   For example, (*extract-digit* 1234 0) should produce 4 and (*extract-digit* 1234 1) should produce 3. You can assume that n has strictly more than k digits.

   Hint: Some built-in Racket functions which you may find useful (though you don't necessarily need to use them) include *expt*, *quotient*, and *remainder*.

   (c) Write a Racket function *check-digit* which consumes a four-digit (i.e., between 1000 and 9999) number *acct-num* and produces a check digit according to the following rules:

   - Add the 0th digit and 2nd digit of *acct-num* together and multiply by 3.
   - Add the 1st digit and 3rd digit of *acct-num* to the above result.
   - The check digit is the remainder of the above result divided by 10.

   For example, (*check-digit* 1234) should produce 2, since:

   - The sum of the 0th and 2nd digits (4 and 2) multiplied by 3 is 18.
   - The sum of the 1st digit and 3rd digits (3 and 1) added to 18 is 22.
   - The remainder of 22 divided by 10 is 2, so the check digit is 2.

   (d) Finally, write the Racket function *add-check-digit* which consumes a four-digit (i.e., between 1000 and 9999) number *acct-num* and produces the account number with a check digit added to the far right.

   For example, (*add-check-digit* 1234) should produce 12342.