CS 115 – Fall 2017
Assignment 02
Due: Wednesday, September 27 at 10:00 a.m.

• Do not use any conditional expressions (cond) on this assignment.
• For full marks, it is not sufficient to have a correct program. Be sure to follow all the steps of the design recipe. Read the Style Guide carefully to ensure that you are following the proper conventions. In addition, your solution must include the definition of constants and helper functions where appropriate.
• Do not copy the purpose directly from the assignment description. The purpose should be written in your own words and include references to the parameter names of your functions.
• The solutions you submit must be entirely your own work. Do not look up either full or partial solutions on the Internet or in printed sources.
• You may post general assignment questions using the discussion groups on Waterloo LEARN. Choose Connect→Discussions. Read the guidelines for posting questions. Do NOT post any code as part of your questions.

• Unless otherwise indicated in the question you may use only the built-in functions and special forms introduced in the lecture slides from CS115 up to and including the modules covered by this assignment. A list of functions described in each module of the lecture slides can be found at https://www.student.cs.uwaterloo.ca/~cs115/built_in
• Download the interface file from the course web page to ensure that all function names are spelled correctly, and each function has the correct number and order of parameters.
• Read each question carefully for restrictions.
• Test data for all questions will always meet the stated assumptions for consumed values.
• Do not send any code files by email to your instructors or any other course staff. Course staff will not accept it as an assignment submission. Course staff will not debug code emailed to them.
• Check MarkUs and your basic test results to ensure that your files were properly submitted. In most cases, solutions that do not pass the basic tests will not receive any correctness marks. Passing the basic tests only guarantees that your solutions can be automatically checked for correctness. It does not mean your solution is guaranteed to be correct for all test cases.
• Read the course web page for more information on assignment policies and how to organize and submit your work. Follow the instructions in the Style Guide.
• Your solutions should be placed in files a02qY.rkt, where Y is a value from 1 to 3.

Language level: Beginning Student
Coverage: Modules 1 and 2

Refer to the String documentation found at: https://www.student.cs.uwaterloo.ca/~cs115/resources/strings.pdf for helpful Racket functions.
1. For this question assume there are 30 days in each month, 12 months in a year and thus 360 days in a year. Complete the function `days-on-earth` which consumes 6 parameters:
   - `byear`, the birth year, assumed to be greater than or equal to 0
   - `bmonth`, the birth month, a natural number in the range 1-12
   - `bday`, the day of the month when born, also a natural number in the range 1-30
   - `cyear`, the current year to check against (i.e. it doesn't have to be 2017), assumed greater than or equal to `byear`
   - `cmonth`, the current month to check against, a natural number in the range 1-12
   - `cday`, the current day to check against, a natural number in the range 1-30

and will produce the number of days on earth for a person born on `byear/bmonth/bday`
(according to the 30 days per month assumption).

You may assume that the date represented by `cyear/cmonth/cday` is always later than or equal to `byear/bmonth/bday`. Also assume the current day is not "finished" so it does not get included in the calculation.

For example:
```
(days-on-earth 0 6 1 0 6 1) => 0
(days-on-earth 0 6 1 0 7 1) => 30
(days-on-earth 0 9 1 1 10 1) => 390
(days-on-earth 0 6 1 1 2 1) => 240
(days-on-earth 1990 6 28 1990 7 2) => 4
```

2. Let's say Timbits can be purchased for $0.25 each, $3 for 20 or $8 for 60. Suppose your instructor wants to buy your class exactly the right number of Timbits so each student gets one and there are none left over. Complete the function `exact-cost` that consumes a natural number `class-size` and produces the lowest cost to buy the exact number of Timbits.

For example,
```
(exact-cost 15) => 3.75
    even though it would have been cheaper to buy 20 for $3
(exact-cost 33) => 6.25
    even though it would have been cheaper to buy 40 for $6. In this case, to get exactly 33 Timbits we buy 20 for $3 and then the remaining 13 for $0.25. This is cheaper than buying all 33 for $0.25.
(exact-cost 170) => 24.5
    In this case we buy 2 packs of 60, 2 packs of 20 and then another 10 individual Timbits
(exact-cost 180) => 24
```

Remember the idea of buying exactly `class-size` number of Timbits is paramount.
3. Complete the function `overwrite` which will consume three strings `s`, `beg` and `end`. The function will produce a string that is a combination of the 3 strings such that `s` will be overwritten with `beg` and `end` placed at the beginning and end respectively. For example:

(overwrite "catapult" "day" "go") => "dayapugo"
(overwrite "catapult" "" "lug") => "cataplug"
(overwrite "" "a" "b") => "ab"

Note that `beg` and `end` will always be present in full in the produced string whereas characters from `s` may or may not be present depending on whether `s` has enough characters to be overwritten. For example:

(overwrite "catapult" "magic" "hat") => "magichat"
(overwrite "catapult" "gogo" "gadget") => "gogogadget"
(overwrite "catapult" "free for all" "!!!") => "free for all!!!"
(overwrite "catapult" "x" "electricity") => "xelectricity"

4. For this question, you will perform step-by-step evaluations of Racket programs, by applying substitution rules until you either arrive at a final value or you cannot continue. You will use an online evaluation tool that we have created for this purpose. You do not need to hand in any files for this question.

To begin, visit this webpage:

https://www.student.cs.uwaterloo.ca/~cs115/stepping

Note: the use of https is important; that is, the system will not work if you omit the s. This link can also be found on the CS115 course webpage, under the Assignments heading.

You will need to authenticate yourself using your Quest/WatIAM ID and password. Once you are logged in, try the “Warmup questions” under “CS 115 Assignment 2,” in order to get used to the system. Note the “Show instructions” link at the bottom of each problem. Read the instructions before attempting a question! When you are ready, complete the four stepping problems in the “Assignment 2 questions” category, using the semantics given in class for Beginning Student. You can re-enter a step as many times as necessary until you get it right, so keep trying until you completely finish every question. All you have to do is complete the questions online—we will be recording your answers as you go, and there is no file to submit. The basic tests for this assignment will tell you whether or not we have a record of your completion of the stepper problems.

Note however that you are not done with a question until you see the message Question complete! You should see this once you have arrived at a final value and clicked on “simplest form” (or “Error,” depending on the question). You should not use DrRacket’s Stepper to help you with this question for several reasons. First, as mentioned in class, DrRacket’s evaluation rules are slightly different from the ones presented in class, but we need you to use the evaluation rules presented in class. Second, in an exam situation, you will not have DrRacket’s Stepper to help you, and there will definitely be step-by-step evaluation questions on at least one of the exams.