Assignment 5
Due at 9:00 AM on Wednesday, November 7

Assignment Guidelines:

- 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.
- Do not copy the purpose directly from the assignment description. The purpose should be written in your own words and include reference to the parameter names of your functions.
- Do not send any code files by email to your instructors or tutors. It will not be accepted by course staff as an assignment submission.
- Read the course Web page for more information on assignment policies and how to organize and submit your work.
- Download the interface file from the course Web page.
- Follow the instructions in the style guide (see the updated version posted on the course web page). Specifically, your solutions should be placed in files a05qY.rkt, where Y is a value from 1 to 4.
- For full marks, it is not sufficient to have a correct program. Be sure to follow all the steps of the design recipe, including the definition of constants and helper functions where appropriate.
- Test data for all questions will always meet the stated assumptions for consumed values.
- Read each question carefully for restrictions.
- Use only material from Modules 1 through 6 in your solution. In particular, do not use list abbreviations.

Language level: Beginning Student.
Coverage: Modules 5 and 6.
Useful structure and data definitions:

(*define-struct fullname* (*first middle last*))
;; A **fullname** is a structure (*make-fullname f m l*), where
;; *f* is a string (first name)
;; *m* is a string (middle name)
;; *l* is a string (last name)

(*define-struct student* (*name term term-grade*))
;; A **student** is a structure (*make-student n t tg*), where
;; *n* is a fullname
;; *t* is a symbol from the set '1a, '1b, ..., '5a, and '5b (latest completed term), and
;; *tg* is a number between 0 and 100 (student's average for the latest completed term)

Constant definitions used in the examples below:

(**define** *name1* (*make-fullname* "Peter" "Lowenbrau" "Griffin"))
(**define** *name2* (*make-fullname* "Agnes" "Gonxha" "Bojaxhiu"))
(**define** *name3* (*make-fullname* "Jerome" "Allen" "Seinfeld"))
(**define** *student1* (*make-student name1* '1a 32))
(**define** *student2* (*make-student name2* '5a 99.9))
(**define** *student3* (*make-student name3* '5a 75.5))
(**define** *student-list* (*cons student1* (*cons student2* (*cons student3 empty*))))

1. The university is conducting a study about students' performance throughout their studies. They plan to look at students' grades in different terms to determine when students experience the most difficulties and receive low grades, and when students' performance is strongest. Write a function *term-grade-stats* that consumes a list *alostud* of *student*s and a symbol *aterm*, and produces the average of term-grades for all students whose latest completed term was *aterm*. If no students just completed *aterm*, your function should produce 'unknown. For example, (*term-grade-stats student-list* '5a) will produce 87.7 and (*term-grade-stats student-list* '2b) will produce 'unknown.
   Hint: you may want to start with a helper function that consumes a list *alostud* of *student*s and a symbol *aterm*, and produces the number of students who just completed term *aterm*.

2. In Assignment 4 you had to write a function *moving-order* to move students into the newly finished residence building "Waterloo Nano". In this question you are asked to solve the same problem but the input is no longer a list of student surnames (strings) but a list of *student*s. The question is restated with appropriate changes here for your convenience:

   Some students are about to move into the newly finished residence building "WaterlooNano". The plan is to move all students with surname initial A to M before others. Write a function *moving-order* that consumes a list *alostud* of *student*s and produces a new list of *student*s that has all students with surnames starting with A to M followed by all the students with surnames starting with N to Z. Note that in the produced list, the students with names starting with A to M remain in the same order as in the original list *alostud*, as do the students with names starting with N to Z. All student names start with a capital letter.

3. Write a function *list-of-squares* that consumes a natural number n and produces a list of perfect squares between 1 and n (inclusive) in descending order. For example, (*list-of-squares* 10) will produce (*cons* 9 (*cons* 4 (*cons* 1 *empty*))). If n is 0, the function should produce an empty list.

4. Write a function *sort-students* that consumes a list *alostud* of *student*s, and produces a new list of *student*s that has all students listed in alphabetical order by their last name. For example, (*sort-students empty*) will produce *empty*, and (*sort-students student-list*) will produce:

```
(cons (make-student (make-fullname "Agnes" "Gonxha" "Bojaxhiu") '5a 99.9)
      (cons (make-student (make-fullname "Peter" "Lowenbrau" "Griffin") '1a 32)
            (cons (make-student (make-fullname "Jerome" "Allen" "Seinfeld") '5a 75.5) empty)))
```