Post Mortem
CS 135 Assignment 4

Design Recipe

Purpose
— Remember to have a purpose for helper functions.
— Do not forget to include parameters in the header of the purpose.

Contract
— Make sure to use brackets around listof ... in contracts.
— Types should be capitalized.
— Write Str for strings, Bool for booleans, and (listof ...) for lists. Note the abbreviation and capitalization.
— Helper functions need contracts.

Examples
— Helper functions need at least one example.

Question 2

in-range
— in-range returns a Nat.
— Remember that (< b a) is possible. Checking only (<= a (first lst) b) is not sufficient.
— Many students knew when to add one, but many had their else statement return 0 when you are supposed to apply in-range on the rest of the list.

spread
— max and min deal with non-empty lists, so the base case is checking if the list’s length is 1.
— Some people sorted the list then found the difference between the last and first element. This is overkill and not recommended, since you have to write a lot more.
— Do not forget in the contract to specify a non-empty list.

Question 3

contains-hot-dog?
— The majority of the students did very well on this question in terms of using simple recursion with a small number of cond cases.
— Many students had contract violations in the functions symbol=? and first.
spells-hot-dog?

—— The purpose should have included "hot dog" or otherwise indicated that "hot dog" is a string.

—— Many students used too many cond cases by defining redundant helper functions. For instance, instead of making a separate helper function to check for each character, make one helper function and have a parameter that contains the character.

Question 4

Template

—— The contract should have contained the data definition for (pair-listof X).

—— Many students had an incorrect base case, either by forgetting to check if the rest of the list was empty or by forgetting to process (... (first plox) ...).

—— Since the data definition states that we are processing two items at a time, the recursive call should have included (rest (rest plox)).

sarcasm

—— Too many uses of string->list and list->string within the recursion.

—— Some students forgot to check when the list is empty after using string->list.

—— For students who followed the template to create their function, the second base case often produced the wrong value.

Question 5

—— Overall, this question was done very well.

—— Checking (zero? (remainder start (* fizz buzz))) will not work for ‘honk all of the time! This is equivalent to (and (zero? (remainder start fizz)) (zero? (remainder start buzz))) if and only if fizz and buzz are coprime. For example, the check fails if start, fizz, and buzz equal 2.

—— A correct contract for fizz-buzz looks like

;; fizz-buzz: Int Int Nat Nat -> (listof (anyof Int Sym))
;; Requires: start <= end, fizz, buzz are positive and not equal

Notice, in particular, the format, that start and end are of type Int, that fizz and buzz are of type Nat, and that all of these requirements are necessary!

Question 6

nat->list

—— When going one step closer to the base case, think about unwrapping the natural number. The natural way to approach this question is to subtract the first digit and then divide by 10.

—— The base case can be tricky to figure out, but it is when the number you are working with is a Digit (that is, between 0 and 9). Think about the last step when considering what the base case should be.

list->nat

—— Some students did not use simple recursion. Remember to use the recursive call of the function to your advantage: the purpose of the function will tell you what it does.

—— Do not forget in the contract to specify that the list is non-empty.