• The non-programming exercises on this homework are due at the beginning of class on Tuesday, May 8.

• For the programming part of this assignment, you should electronically submit your definitions and interactions files by 11:59pm on May 7, and turn in the requested printouts by the beginning of class on May 8.

• When writing up your homework, please explain your answers clearly and write neatly. Unexplained answers may not receive full credit.

Non-Programming Exercises

In each exercise asking for a probability below, be sure to specify the particular expression being evaluated in terms of the relevant events (e.g., “p(E), where event E is . . . ” or “p(E | F), where . . . ”), as well as giving the calculated probability.

Also, on page 151 of your Makinson textbook, Exercise 6.4.2(3).b establishes the following fact about conditional probability:

**Theorem:** For any events A, B, whenever p(B) ≠ 0, p(A | B) = 1 − p(A | B).

In these exercises, you may use that fact without proof. (As always, however, you must cite it explicitly if it is used!)

1. Annika, Bobby, and Nancy are each playing a round of miniature golf. From their past records, we know that their probability of finishing the round with a score under par is $\frac{1}{4}$ for Annika, $\frac{1}{5}$ for Bobby, and $\frac{1}{6}$ for Nancy. We assume that their performances are independent of each other. If we know that exactly one of them finishes the round with a score under par, what is the probability that Bobby finished under par?

   (A complete answer for this exercise will include specifications of the relevant sample space and the events necessary to model the question as asked.)

2. *(This exercise is part of Makinson, ex. 6.5(b).)* At any one time, approximately 3% of drivers have a blood alcohol level over the legal limit. About 98% of those over the limit react positively on a breath test, but 7% of those not over the limit also react positively. Find the probability that an arbitrarily chosen driver is not over the limit given that the breath test is negative.

Programming Exercises—Random Deals and Probabilities!

These exercises are about random card deals and probabilities. Be sure to download the following files into your working directory for this assignment:

• asmt-helper.txt
• hw7-helper.txt

In your Scheme program, you only need to load hw7-helper.txt—it takes care of loading asmt-helper.txt.

NOTE: Even though the examples in these instructions do not show the use of tester function expressions, please be sure to have adequate tester expressions in all of your examples!

1. Make A Full House! Define a function, called rand-full-house, that takes no inputs. It should generate as its output a list of five cards (i.e., numbers from 0 to 51) that has a “full house” (i.e., 3 cards of one rank plus two cards of another rank). This function must not play favorites! Each hand that has a full house should be equally likely.

One way of implementing this function is to repeatedly deal five-card hands until one shows up with a full house—but not for this exercise! (That approach is extremely inefficient!)

For this exercise, take an approach that’s related to one of the ways we computed the probability of a full house: consider the number of ways of choosing ranks and suits. Here, however, except that instead of determining the number of choices, you’ll actually make a random choice. That is, instead of analysis based on the idea that the rank of a card could be any one of 13 numbers, randomly select one of the 13 numbers, and proceed from there!

NOTE: Functions in hw7-helper.txt (e.g., cards->string) should be helpful!

   > (cards->string (rand-full-house))  
      "6C 6S 6D 2D 2H "  
   > (cards->string (rand-full-house))  
      "10S 10H 10D 3H 3D "

2. Even Distribution! Let E be the event that a hand of five cards has at least one card in each suit. Define a function has-even-distn? that takes a list of cards (i.e., numbers from 0 to 51) as its only input and returns #t if those cards have at least one of each suit (and returns #f otherwise). (HINT: The suit function could come in handy!) Here is some example output:

   > (define cards1 (rand-poker-hand))  
   > (cards->string cards1)  
      "AS 2S 8C 2H 6H "  
   > (has-even-distn? cards1)  
      #f  
   > (define cards1 (rand-poker-hand))  
   > (cards->string cards1)  
      "KC 6S AS 9D 8H "

2
3. **Computing some probabilities!** Let $E$ be the event that a hand of five cards has even distribution (i.e., at least one card from each suit), as above. Also, let $F$ be the event that a hand of five cards has a full house, and let $p$ stand for our probability function.

Define the following global variables:

*prob-even-distn* $\leftarrow p(E)$
*prob-full-house* $\leftarrow p(F)$
*prob-even-distn-and-full-house* $\leftarrow p(E \text{ INTERSECT } F)$
*prob-even-distn-given-full-house* $\leftarrow p(E \text{ GIVEN } F) = p(E|F)$
*prob-full-house-given-even-distn* $\leftarrow p(F \text{ GIVEN } E) = p(F|E)$

Recall from conditional probability that

\[
p(F|E) = \frac{p(F \cap E)}{p(E)}
\]
\[
p(E|F) = \frac{p(E \cap F)}{p(F)}
\]

If you know two of the probabilities in either of these equations, you can solve for the third. For this exercise, some of those probabilities are easier than others to compute directly (rather than solve for, from the other two in an equation). **HINT:** If you get stuck on one or more, continue to exercise 4, which might provide some clues.

We already know that $p(F)$ is about 0.00144; for this exercise, you should first compute the numbers of full houses and five-card hands using Scheme, and then use those constants to compute the value of $p(F)$. Follow a similar procedure to compute the probability $p(E)$, giving the Scheme expressions that calculate the relevant values. For the conditional probabilities, please also give the Scheme expressions that calculate the requested values: Do not simply type in the probabilities as numbers and have your code return them; instead, you should write code that does the relevant calculations.

**NOTE:** Be sure to use `tester` expressions (e.g., `(tester 'prob-full-house*)`) to display the values of these variables! Also, use `tester` to show the value of $p(E)p(F)$—this should not be the same as $p(E \cap F)$, because these two events are not independent!

As in our lab exercises, all probabilities should be returned as decimals, not ratios of integers.

As always, please explain your answers—here, please be sure that comments in your code contain explanations of your answers!

4. **Using gen-and-test!** Use `gen-and-test`, where `rand-full-house` is the generator function, and `has-even-distn?` is the testing function. In comments for this function, please answer the following questions:

```scheme
> (has-even-distn? cards1)
#t
```
(a) Which of the above probabilities should this estimate?
(b) What is the value for that probability?

As always, please explain your answers—here, please be sure that comments in your code contain explanations of your answers!

Notes and Submission Instructions

To submit this exercise, please electronically submit a directory containing your definitions file, your interactions file, and the helper files, unaltered from the ones given to you with this HW. Your definitions and interactions files must both be plain text files whose names contain your name and end with the “txt” extension (e.g., eaaron-hw7-defns.txt). In addition, you must turn in printouts of those two files.

Additional instructions and points to emphasize (many of which are repeated from the previous HW):

- Before submitting your code, make sure that your code is properly indented and adequately commented! As part of that, make sure that each function has comments that specify the name of the function, its inputs, and its outputs. (Such comments could be called a contract.) Also as part of that, make sure that there is a blank line between the contract and the function definition, and make sure that there are no lines of text that “wrap around” to the next line, either when printed or when viewed on an 80-column screen. These are important points of readability for good coding practices!

- Moreover, if code contains multiple functions that are intended to be used together (e.g., a helper function along with a main function), comments should indicate that connection as part of describing the overall functionality of the code.

- When submitting your answer, be sure to include a thorough collection of tester expressions to demonstrate that your functions are working properly. It is not sufficient to simply include the test cases given in these instructions.

- Please make sure that you have saved your interactions as text!

- Any functions you define for this assignment should be in your definitions file.

- For Paper Submissions: Print each file separately—that way, you won’t get one file starting on the back of the other. Staple the definitions printout on top of the interactions printout. (Note: Your non-programming exercises can be turned in separately, i.e., not stapled to the program files.)

Not following the above instructions and guidelines may result in point deductions—good style is essential to good coding!—so please do follow the instructions!