Permutations / C programming ungraded homework

This homework is ungraded. Its purpose is to help you understand course material. Doing these problems may help you with your graded homework and quiz. Problems like these may appear on the midterm or final. You should attempt to complete them before the end of the module (time permitting), and at a minimum study them and their provided solutions.

1) Let Z_n be shorthand for the set containing the n smallest non-negative integers (eg, $Z_3 = \{0,1,2\}$). Is f: $Z_5 \rightarrow Z_5$ defined as f(x) = 2x mod 5 an invertible function? If so, what is its inverse (given either as a set of ordered pairs or as a formula). If not, explain in one short sentence. *Note: since this signature is of the form A → A, if it is invertible then it can also be called a permutation or permutation function.*

2) (a) How many functions exist with signature f: $Z_4 \rightarrow Z_5$? (b) Given that a and b are positive integers, how many functions exist with signature f: $Z_a \rightarrow Z_b$? (c) How many permutation functions exist with signature f: $Z_4 \rightarrow Z_5$? (d) How many permutation functions exist with signature f: $Z_4 \rightarrow Z_4$? (e) Given that a and b are positive integers, how many permutation functions exist with signature f: $Z_a \rightarrow Z_b$?

3) Let `rand(n)` be a library function that evaluates to a random integer in Z_n each time it is called (like Java's `Random.nextInt(n)`). Write a method called `createRandomFunction` (right here in your homework) in C, Java, or pseudocode that takes a positive integer n as a parameter and returns an array with n elements each uniformly distributed in Z_n. Essentially I'm asking you to write a method that specifies a random function $Z_n \rightarrow Z_n$ using the table filling method (ie, `a = createRandomFunction(10)` fills `a` with random values and then `a[0]` would tell you what 0 maps to, `a[1]` tells you what 1 maps to, etc.).

4) Do Problem 4 again, but this time name the method `createRandomPermutation` and make the array a permutation (ie, 0 through n-1 each appear exactly once). For full credit, make your method run in O(n) time.

5) Let a and b be integers greater than 2, and let f: $Z_a \rightarrow Z_b$ be a random function. (a) What is Pr[f(0)=0]? Explain. (b) What is Pr[f(1)=1 | f(0)=0]? Explain. (c) What is Pr[f(1)=0 | f(0)=0]? Explain. Let a be an integer greater than 2, and let f: $Z_a \rightarrow Z_a$ be a random permutation function. (d) What is Pr[f(0)=0]? Explain. (e) What is Pr[f(1)=1 | f(0)=0]? Explain. (f) What is Pr[f(1)=0 | f(0)=0]? Explain.

6) A `string` is a concatenation of symbols from an alphabet. For example a string of bits is a concatenation of elements from the alphabet {0,1} and a string of bytes is a concatenation of elements from the alphabet {x | x is an 8-bit string}. A String in Java is the concatenation of elements from the type `char`. (a) Let A be the set of all bytes (ie, all 8-bit strings). How many elements are in A? Explain. (b) Let B be the set of all 8-byte strings (ie, all 64-bit strings). How many elements are in B? Explain.

7) You are given a black box *f* that has either a standard 52-card deck-of-cards or a 48-card deck-of-cards for the game pinochle. You are allowed to activate *f* once, upon which a card is chosen at random and you are given the card. In pseudocode, give an algorithm that uses *f* once and then guesses either "standard" or "pinochle". Evaluate the advantage your algorithm achieves.

In this problem you can maximize your advantage by identifying all of the cards that are more likely with one type of deck and guess that type of deck if any of those cards appear.

8) You are given a black box *f* that has either a 30-sided die or a 34-sided die inside. Each time you activate *f* the die is rolled and you are told the resulting value. You are allowed *q* activations. In pseudocode, give an algorithm that uses *f q* times and then guesses either "30-sides" or "34-sides". Evaluate the advantage your algorithm achieves as a function of *q*.

9) At https://codestepbystep.com do the following problems in the C/bitwise section: ascii_to_hex, binary_to_hex1, bitmask0, bitmask1, bitmask2, bitmask3, bitset1, bitwise1, hex_to_binary1.

Ungraded homework solutions

These are best studied after completing the homework or after struggling with it for a while.

Problems 1-8: solutions.
Problem 9: solutions.