_____ Test ID Number

_____ Student ID Num

ID Sheet: Write your seven-digit BYUH Student ID number in the blank above. Turn in this sheet when you complete the test. It will be kept separate until grading is completed, and will then be used to assign your score to the proper person.

The "In-Class Test Rules" provided herewith apply to this exam.

On each of the problem sheets, write your Test ID Number in the small box in the upper left corner of the page. Then perform the assigned task (for example, write a program) in the big box. DO NOT WRITE YOUR STUDENT ID NUMBER OR NAME ON ANY OTHER TEST SHEET.

Write a normal, interactive program unless a CGI program is requested.

There is a PERL CGI module that exists. Don't use it.

For database problems, if it is not specified: assume the database is "DDD"; assume the hostname is "HHH"; assume the password is "PPP"; assume the table is "TTT"; assume the user is "UUU". Never mention your own database, user, or password.

Ending the Test Generally I will warn you as the test is coming to a close. I may state "Ten Minutes Remaining," "Five Minutes Remaining," "Two Minutes Remaining," "One Minute Remaining," and "Put down your pencils." I may instruct you to leave your papers neatly arranged on your desk for me to collect after you leave the room.

If you keep writing after I instruct you to put down your pencils, your test may be refused, or I may deduct points from your score.

Turning In Your Test If the pages of the test are numbered, put them in the order of those numbers.

If the pages of the test are **not** numbered, put the "In-Class Test Rules" on top. Put this sheet second. Put the individual problem solutions next, in order by problem number. Put any remaining sheets last.

The key concept is to prevent me from seeing or memorizing your test ID number, as that could damage my ability to grade anonymously.

Grading

Problems will be graded on the following scale:

Points	Descriptive Rubric
20	perfect or small mistake
17	one medium mistake
15	two medium mistakes
13	one large mistake
10	half right, several mistakes
0-9	less than half right

Points are awarded for achieving the major goal of the problem. Points are not awarded for merely providing incidental details without making substantial progress toward the major goal.

Read each problem carefully. Make your program do everything that is requested. Avoid doing anything that is not requested. Identify any special assumptions you are making.

Points can be lost for including extraneous work, as this suggests one does not know what is needed, and one is simply throwing in whatever comes to mind in hopes that some of it is right. Lines randomly copied from the hints sheet will not get you points.

Points can be lost for presenting a correct solution that is substantially less efficient than the desired solution. In particular, the use of unnecessary loops can cost points.

There are six problems. Solve any five. Cross out the one I should not grade. Turn in all six sheets.

Test ID Number

1 Ducks

A carnival booth has two tubs of water. One tub is yellow. One tub is orange. In each tub a number of plastic ducks are floating in the water. The ducks appear to be identical, but in fact each duck has a number written on the bottom where it cannot be seen.

In a contest of luck, one player draws a duck from the yellow tub and another player draws a duck from the orange tub. The player with the higher number wins a prize.

You are given the numbers from the bottoms of the ducks preloaded in two global arrays: @yellow and @orange. Write a subroutine to calculate and print which tub should win more often on average.

Test ID Number

2 Monte Carlo

"Roll one die" means generate a random integer, 1, 2, 3, 4, 5, or 6, with equal probability.

Write a program to roll one die ten thousand times. Print how many times each number was rolled. For example, "1 occurred 1607 times", "2 occurred 1723 times", etc.

Test ID Number

3 Vote Counter

Write a CGI program. Each time it runs, it lists two candidate names (make up something funny), each on a submit button, and the total votes cast for each.

The first time it runs, set the vote totals to zero. When a submit button is pressed, the same program should run, update the total votes for that candidate, and display the updated screen. Maintain the vote totals in one or more hidden fields.

Test ID Number

4 Valid Numbers

Write a program that accepts one line of input and correctly prints "VALID" or "NOT." A valid number can have a plus or minus in front (but no place else). It must have one or more digits. It can have a decimal point. Leading zeros are not allowed unless zero is the only digit before the decimal.

Here are some examples of valid numbers: `0`, `1`, `11`, `1.000`, `0.1`, `.1`, `1.`, `+12`, `-12`, `+.12`, `-.12`, `293.3467`, `12.34`, `+12.34`, `-12.34`

Here are some examples of invalid numbers: `007`, `1..3`, `1.2.`, `1.2-`, `12,345.67`, `293-3467`, `808.293.3467`, `2-`, `--2`

Test ID Number

5 Select Star

Write a perl subroutine named sstar. As input, it is given a table name. As output, it prints the entire contents of that database table (select * ...) in HTML table format (using `<table>` `<th>` `<tr>` `<td>` etc.). For extra credit, include name and data type for each column.

Test ID Number

6 Verify Table

Write a CGI program that asks for a table name. When the table name is entered, it connects to a database and verifies whether that table exists. It then correctly reports "`$table` EXISTS" or "`$table` DOES NOT EXIST" and again asks for a table name.