Comprehensive Final Exam for Capacity Planning (CIS 4930/6930) Fall 2001

>>> SOLUTIONS <<<

Welcome to the comprehensive final exam in *Capacity Planning* (CIS 4930/6930). You have 120 minutes. Read each problem carefully. There are ten required problems (each worth 10 points) and one extra credit problem worth 5 points. You may have with you a calculator, pencils, erasers, blank paper, lucky rabbit's foot, and one 8.5 x 11 inch "formula sheet". On this formula sheet you may have anything you want (definitions, formulas, etc.) *handwritten by you*. You may use both sides. Please start each numbered problem on a new sheet of paper and do not write on the back of the sheets. No sharing of calculators.

Problem #1

Answer the following general questions about capacity planning. One or two words will suffice for some of the questions. No more than one or two sentences for the remaining questions.

a) As a first-order definition, what is "capacity" in the context of computer systems?

Capacity = throughput.

b) What is "performance"? Describe the measures of interest.

Performance = throughput and delay. Also utilization, loss, reliability, and availability.

c) Describe the capacity planning process (hint: what are the inputs and outputs to the process?)

Capacity planning is a human endeavor that uses performance evaluation. Inputs are workload evolution, system parameters, and desired service level. Outputs are saturation point (i.e., where desired service level can no longer be met given the specified workload and system) and alternatives.

d) What one thing dominates performance issues in computer systems?

Memory speed << CPU speed.

e) Describe key performance issues in the design of a CPU.

A key performance issue in a CPU is to trade-off cache size for more complexity in the CPU instruction unit (e.g., parallel processing of instruction streams). Other issues include die size and development time.

f) Give the generic Utilization (U) formula that applies to virtually all computer and communications systems.

U = T_useful / (T_useful + T_overhead)

Problem #2 (Note: A T-score table is at the end of the exam)

You are given the following measurement values…

```
101, 102, 101, 103, 102, 101, 105, 107, 104, 124
```

a) Assume that the measurements are a population. Find the mean, variance, standard deviation, and CoV.

$$E[X] = \frac{1}{10}(101+102+101+103+102+101+105+107+104+124) = 105$$

$$\sigma^2 = \frac{1}{10}\left((101-105)^2 + (102-105)^2 + \cdots + (127-105)^2\right) = 43.6 \qquad \sigma = 6.603 \qquad CoV = \frac{\sigma}{E[X]} = 0.0629$$

b) Plot a histogram of the measurements.

```
% occurrence |
             |                      10% of values beyond
             |                      the range of this plot
        30%  |   |                  (1 value of 124).
        20   |   |   |
        10%  |   |   |   |   |   |       |
             +---+---+---+---+---+---+---+---+--- value
                101 102 103 104 105 106 107
```

c) Assume that the measurements are a sample. With 95% confidence, give the interval within which the population mean lies.

We need to first compute the population standard deviation,

$$s = \sqrt{\frac{1}{10-1}\left((101-105)^2 + (102-105)^2 + \cdots + (127-105)^2\right)} = 6.96$$

$H = 2.26 \cdot \left(\frac{6.96}{\sqrt{10}}\right) = 4.974$ (we use 2.26 for the T score for N - 1 and $\alpha/2$ = 0.025)

Thus, the true population mean lies between 100.03 and 109.97 with 95% confidence.

Problem #3

You have been given a function with prototype `int compute_answer(void)`. A typical execution path in this function is several hundreds of lines of C code. The function takes no input and returns an integer value. Show (e.g., write a code snippet) how the execution time of this function could be measured.

```
#include <stdio.h>          // Needed for printf()
#include <sys\timeb.h>      // Needed for ftime() and timeb structure
int computer_answer(void);  // Prototype
void main(void)
{
  struct timeb start, stop; // Start and stop times structures
  double       elapsed;     // Elapsed time in seconds
  int          num_times;   // Number of times to invoke test function
  int          i;           // Loop counter
  // Start timing and then invoke the function num_times
  ftime(&start);
  num_times = 100000;
  for (i=0; i<num_times; i++)
    compute_answer();
  // Stop timing, compute time per function execution, and output
  ftime(&stop);
  elapsed=((double) stop.time + ((double) stop.millitm * 0.001)) -
          ((double) start.time + ((double) start.millitm * 0.001));
  printf("  Time per function execution = %f millisec \n",
    1000.0 * elapsed / num_times);
}
```

Problem #4

Describe what is MRTG and how it "works". Details matter.

MRTG is Multi-Router Traffic Grapher. MRTG uses a Perl script to GET SNMP variables (typically, bytes in and out of a router port) and then plots them on a graph. The graph is inserted into a webpage which is stored in a preset directory. If MRTG is run on a webserver, the web pages are then viewable from anywhere. A key feature of MRTG is its ability to compress graphs from a minute, to hourly to daily to weekly and so on views. This compression (achieved by blocking) makes archiving of data not require ever increasing storage space. MRTG is freeware and runs on Unix and Windows platforms. MRTG was written by Tobias Oetiker.

Problem #5

Write a C function that returns a file size (in Kbytes) from an empirical distribution based on the below measurements of 1200 files in a file system (is it believed that these 1200 file sizes are representative of the file system).

File size	Number of files
1.0 Kbytes	100
3.5	200
4.0	500
5.0	300
10.0	100

```
double emp(void)
{
  double z;    // unif(0,1) RV
  z = (double) rand() / RAND_MAX;
  if (z <= (100.0 / 1200)) return(1.0);
  else if (z <= (300.0 / 1200)) return(3.5);
  else if (z <= (800.0 / 1200)) return(4.0);
  else if (z <= (1100.0 / 1200)) return(5.0);
  else return(10.0);
}
```

Problem #6

Design a benchmark for a CPU. Describe the workload, measurements to be taken, and how the benchmark can be implemented. Carefully describe what your benchmark measures. Overall, the WHY is much more important than the WHAT. This question tests if you know what a benchmark is and what aspects are of interest to benchmark in a CPU.

A benchmark should be representative of a real workload. For a CPU there should be instruction mixes that execute out of L2 cache and also cause L2 cache misses, that exercise both floating point and integer units, and that exercise any branch look-ahead capabilities (i.e., pipeline processing of future instructions). The instruction mixes should be based on real applications targeted for the CPU being evaluated. The key measurement of interest is program execution time. For simple CPUs without instruction pipelining, it may be possible to reduce this to a simple instruction execution time. The implementation of the benchmark is a set of programs that are executed out of system memory and begin timing only when fully loaded and executing.

Problem #7

Answer the following questions regarding web servers.

a) Give the high-level flowchart showing how a web server works.

```
Repeat the following forever
  1) Wait for a connection
  2) Make the connection
  3) Receive and parse an incoming HTTP request
  4) Handle the HTTP request (if a GET, send the requested file)
  5) Close the connection
```

b) Describe what is HTTP and its key attributes. List at least three HTTP commands.

HTTP is HyperText Transfer Protocol and runs over the TCP/IP protocol. HTTP is a request-response protocol between a Web client (typically, a browser) and a web server. HTTP uses ASCII protocol headers. An HTTP GET is used to get objects from a web server which are returned in an HTTP packet with an HTTP response header. HTTP headers include the HTTP command (e.g., GET, PUT, POST).

c) Describe what are the key performance trade-offs in implementing a web server and how can these performance issues be addressed?

The key performance trade-off is the cost of threading or processes for parallel connections and I/O for a connection. Without threading (or processes) parallel connections cannot (easily) be supported. With threading, each connection has overhead to start and stop a thread or process. Caching of frequently accessed (small) objects can reduce I/O time.

Problem #8

Give Amdahl's Law. Describe what the law "means" and give a pictorial (i.e., by pictures or figures) derivation of Amdahl's formula.. Given an example that demonstrates the use of Amdahl's Law.

$Speed-up = \frac{q}{(1+\alpha(q-1))}$ where α is the percentage of application that is serial by nature and q is the number of processors. This means that the maximum possible speed-up of an application is limited by its serial portion. Speed-up is limited to the time it takes to execute the serial portion on one processor.

```
                        T_old
|-----------------------------------------------|   The single processor case

   αT_old          (1 - α)T_old
|----------|------------------------------------|   Where α is serial fraction

    αT_old   (1 - α)T_old/q
|----------|--------------|   Time for q processor case (with linear scaling in
                              non-serial portion
```

Thus, the speed-up is T_old / (aT_old + (1-a)T_old/q) which simplifies to
q / (1 + α(q - 1))

Problem #9

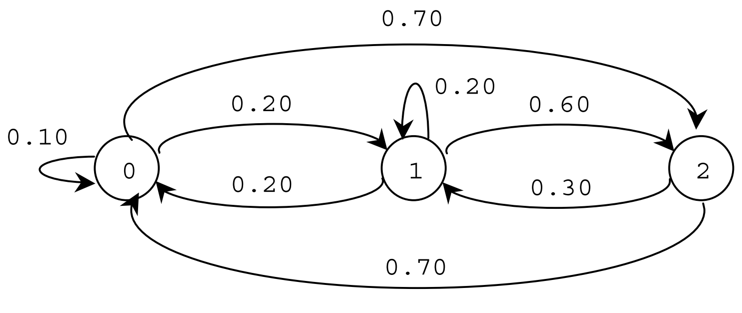
Answer the following questions about Markov modeling.

a) What kind of system can be modeled as a Markov model?

Systems with discrete state than have a memoryless transition between states (whether at discrete intervals in time or continuous in time) can be modeled using Markovian methods.

b) Below is a probability matrix for a discrete time Markovian system. Draw the Markov chain corresponding to this P matrix (number the states as 0, 1, and 2). What is the probability that an outside random observer will find the system in state 1

```
P = | 0.10   0.20   0.70 |
    | 0.20   0.20   0.60 |
    | 0.70   0.30   0.00 |
```



```
-0.90p0 + 0.20p1 + 0.70pw = 0
 0.20p0 - 0.80p1 + 0.20p2 = 0
     p0 +     p1 +     p2 = 1
```

Solving for p1 = 0.24

<u>Problem #10</u>

Answer the following questions regarding simulation modeling.

a) List the common components in a simulation program

`System state, simulation clock, event routines, statistical counters, library routines, report generator, initialization, and main program.`

b) Give the general flow chart or pseudocode for a discrete-event simulation program

```
Initialize global state variables
Initialize global time to zero
Obtain the first input event
While (not yet done)
{
  Remove next event from event list
  Advance time to time of this event
  Perform event routine for the event type of this event
  Update global variables
  Generate next event triggered by this event
}
Generate report
```

c) Describe the event list. What is its purpose? How does it "work"?

`An event list is a` <u>`linked list`</u> `with` <u>`event structures`</u> `that contain, at a minimum, the` <u>`event number and its time of occurrence`</u>`. A simulation program must be able to` <u>`remove the head event`</u> `from the list and` <u>`insert an event`</u> `into an arbitrary location in the list based on a time ordering of the list.`

<u>Extra Credit:</u>

If a population of values (e.g., of file sizes) has an infinite mean and infinite variance, what will the pdf and PDF "look like"? Assume that the majority of the members of the population do have a finite value. Sketch an example pdf and PDF from such a population.

`The distribution will be` <u>`heavy tailed`</u>`.  The PDF will never get to 1.0 (in finite x) and the pdf will never reach 0.0 (in finite x).`

T-scores. Selected values of $t_{\alpha/2;N-1}$

	$\alpha/2 = 0.05$	$\alpha/2 = 0.025$
N - 1	**t**	**t**
4	2.13	2.78
5	2.02	2.57
6	1.94	2.45
7	1.90	2.37
8	1.86	2.31
9	1.83	2.26
10	1.81	2.23
11	1.80	2.20
12	1.78	2.18