**Homework#1 (h1) CSC148, Prof Mitchell Due dates: 1) EXTRA CREDIT: start of class, Tuesday Sept 10 or**

**2) NORMAL DUE DATE: start of class, Thursday Sep 12 Histogram via QTABLE example (at end of doc), added 9/7**

**#1 Introduction to Using gpss**

gpss does not have a large GUI interface, nor does it provide a “project” structure that is so common in many current languages/systems. As the course notes indicate, this course does not emphasize animation/audio interaction with a user. Instead, the major focus of gpss is to model several common cases of service processing.

Later we cover material on models having an exact mathematical solution for the common statistics of interest such as , W, avg(L), etc. Simulations are not needed for common statistics in such systems.

Invoking gpss:

The sequence: File 🡪 New 🡪 Model (NOT Text File) 🡪 OK

invokes an editing session in which you can input a new “model”, a new gpss-formatted source file (*caution: gpss source is not a .txt file*).

Input all source lines to build a complete gpss model. With source file named h1.gps, save using File menu option: Save As. *(.gps is the gpss source file type*)

Instead of creating your source with the above ‘New’ option, you can modify a copy of the barberShop simulation demoed in class last week, located at: http://athena.ecs.csus.edu/~mitchell/csc148/homework/h1BaseModel\_148\_f19.gps)

**#2. Introduction**

This assignment implements significant extensions of the gpss barberShop example:

Part 1. The {ia} and service distribution parameters are changed to ensure a stable model

Part 2. Uniform {ia} and service distributions are compared with exponential distributions with the same distribution means as in 1.

Part 3. Blocks and statements are added to get statistics beyond default statistics.

Part 1 – is needed for model stability

Part 2 - This seemingly simple change radically alters many steady-state (i.e., long-term) statistics. For both arrival and service “processes”, the difference is due to the unequal ia probabilities for exponential distributions.

Recall from your statistics background: the exponential probability mass function (pmf) is defined on (has domain) (0,+∞), and is unbounded, compared with a uniform pmf having domain [a,b], 0 < a < b, a and b positive real numbers.

With an exponential distribution, even though the probability is small that a given cj has a very large service duration, when that does occur, the queue length will correspondingly increase. That is, avg(queue length) is generally much larger compared with uniform distribution. Similarly, it is possible for several consecutive very small (near 0) ia values, thus creating a sudden increase in service wait queue.

*h1 results illustrate the importance of choosing correct/appropriate distributions for all simulation processes (arrivals/services/other actions) when building a DES model.*

*For the rest of this document, abbreviate exponential as “****exp****”*

**#3.** **CSC148 source documentation requirements**

The h1BaseModelxxx source has preamble documentation in 2 parts. You are required to follow the statement and block coding standards of 2. in all gpss programs in this course.

In addition, every gpss program in this course must include the following 4 elements in the preamble documentation (= first lines of each gpss program).

1. Brief 1-2 line functional description of the model; include your name & date
2. t.u. definition (Example t.u. 1 hour)
3. Start parameter (Example Start 1000 would execute 1000 haircuts)
4. Termination condition: for most models, it is tc <- 0 (meaning, the run

terminates when the model tc counter value becomes 0)

gpss files can be stored in any folder, independent of gpss .exe location and the directory that includes it.

Note - gpss File main menu item has the simple sub-menus Open, Close, Save and Save As – all typical of Windows OS windows that interface with files.

While still in an editing session for h1.gps, and assuming you are ready to execute the source file, select gpss menu items

Command 🡪 Create Simulation. This creates a “journal” file (by default, named h1.sim) that traces major states (start, end, suspend, etc.) during simulation execution. If error messages do not appear after issuing Start n, you should see 2 lines indicating successful model translation, like:

07/17/19 16:53:34 Model Translation Begun.

07/17/19 16:53:34 Ready.

gpss now waits for a command (to execute the model, or other command(s))

Choose: Command 🡪 START, and a small popup window contains START 1. The “1” is default initialization for the (termination count (tc)) variable.

Input tc value *5000* for each h1 run. Click OK and run begins.

The above is the basic sequence for running a single gpss simulation. Save/delete, etc. any of the involved files as needed. For example, each time a simulation run is executed, another x.sim journal file is created (an x.sim file logs major events of a model run). A default report (.rpt) file is also generated. It contains many summary stats from model execution; it is a major advantage of this kind of simulator – you do NOT need to code any of the logic for report output; it is generated automatically by gpss.

Since gpss-generated x.gps source files are not raw unformatted .txt files, they should be opened only from within a gpss session; on the other hand, report files and x.sim files are ordinary Windows text files.

gpss language case sensitivity

You can intermix upper and lower case in gpss language tokens for more readability (Example: identifier oneLABEL is the SAME as ONELABEL).

The following test model illustrates this:

; Illustrate label case insensitivity (same is true for blocks and statements)

generate 1

oneLABEL advance 1

ONELABEL advance 1

terminate 1

Translates with errors:

09/03/17 21:04:54 Model Translation Begun.

09/03/17 21:04:54 Line 3, Col 1. Multiple label definitions:

09/03/17 21:04:54 ONELABEL advance 1

09/03/17 21:04:54 \*\*\*\* Model Translation Aborted \*\*\*\*

**#4. Specifications for Extending the single-barber simulation**

*h1 specifications for t.u. are DIFFERENT from first example model SAMPLE1.GPS.*

*The t.u. is 1 minute, not 1 second. Thus, GENERATE and ADVANCE block operand scale must be adjusted compared with the 1 second t.u. for SAMPLE1.GPS.*

*Recall that the time values in gpss block/statement operands, as well as report results, are based on the programmer’s chosen t.u. value.*

=============================================================================

The following specifies how SAMPLE1.GPS model source code is extended.

This doc assumes the server is named Barber (you can name it anything).

**Add** a **CLEAR** statement as the first block or statement following the Preamble documentation. This statement removes any residual or state values (if any) persisting from previous gpss executions in a gpss session.

*Always using CLEAR is a gpss coding convention for rest of course.*

**Add** **EQU** statements (place them after the CLEAR and before any other

statements or blocks) to define a symbol for each numeric or string scalar value. EQU is the way to define a symbol and its value in gpss.

Example EQU statement along with a comment:

interArrMean EQU 10 ; {ia} mean is 10 minutes

(You know the benefits of symbolic code from previous programming experience;

the same applies for gpss).

**Add** an EQU statement for each GENERATE and ADVANCE block operand (all 4 of them).

Using the EQU statement, define GENERATE A and B operand values **10 and 7,** and ADVANCE A and B operand values **7.5 and 2** as symbols, **not** hard-coded integers.

**Add** one **EQU** statement and one **RMULT** statement following this EQU to seed your model’s random number generation as follows:

Label your EQU statement and for its X operand, choose any six-digit unsigned integer seed value. Then code an **RMULT** statement with A operand = the label of the EQU statement. (RMULT will NOT need a label).

The EQU and RMULT cause an internal gpss random number generator to calculate random numbers for interarrival and service durations and other internal runtime values for each simulation run.

**Gathering Statistics**

In general, a QUEUE/DEPART pair can be placed at any positions in source code – the placement depends on what delays/durations must be measured.

**Remove** the existing **QUEUE** and **DEPART** blocks from the BaseModel code. Recall that they gather statistics about service waiting.

*---------------------*

***Notes****: When first using gpss, QUEUE/DEPART block pairs are confusing because of their gpss block entity names. A QUEUE block does not cause the queuing of an entity to some queue (Each SEIZE implements its own internal service wait queue).*

*The purposes of a pair of QUEUE/DEPART blocks are to measure statistics of various kinds as each cj proceeds after QUEUE and to the corresponding DEPART.*

*gpss coding – Although QUEUE and SEIZE in SAMPLE1.GPS have the same operand (Barber), which is legal, it is confusing, and makes a Report file more difficult to interpret = > avoid using the same name for entities of different types.*

*Report Note - when QUEUE/DEPART pairs are not used, a .rpt file will not contain corresponding statistics in the report section with head label QUEUE.*

*A complex model has MANY QUEUE/DEPART block pairs; nested and overlapped pairs are allowed.*

*----------------*

**(the number of cj in S) statistic**

**Add** a **QUEUE And DEPART** with operand **res\_dur**; QUEUE follows the GENERATE and DEPART precedes the TERMINATE.

Corresponding to the QUEUE/DEPART block pair with operand res\_dur

**Add** a **TABLE** statement anywhere in the definitions above the GENERATE block.

In general, TABLE operands specify the histogram of the distribution of some random variable of interest (queue length, service wait time, etc.).

In h1, the table specifies display the distribution of (the number of cj in the barber shop as each cj arrives). This standard stat is denoted in CSC148 by **L**.

The TABLE Name (gpss doc calls it a label) operand can be any meaningful identifier you choose. The A operand must be the gpss built-in function call: **q$res\_dur**; it returns the current L value.

For the operands B, C and D: the first cell’s right-most value is 1 and each cell is size 1. Determine (by doing some runs) operand D to get a histogram where the empty right-end cells are a very small fraction of all cells.

The D operand must be a positive integer (cannot be a symbol).

ReferenceManual chapter 6 has complete documentation for TABLE and its operands.

(chapters 6, 7, and later, some parts of chapters 3 & 8 are needed for CSC148)

**Add,** right after the QUEUE res\_dur block, a block **TABULATE tableName** block**,** where tableName is the label of your TABLE statement**.** This block increments by 1 the histogram cell corresponding to the number of cj in S as cj enters S.

As gpss documentation explains, TABULATE appears at a place in the code at which the statistic for its TABLE statement needs to be collected. This statistic is based on the state of the model at the model time when the TABULATE is entered.

*See the Section below that outlines code for a TABLE & TABULATE statement.*

**Server duration statistic**

**Add** QUEUE and DEPART blocks with operand **svr\_dur** at appropriate places to gather service duration stats. This statistic is easier to obtain. Instead of a TABLE and corresponding TABULATE, only a QTABLE statement is needed. As with TABLE, QTABLE has a label, but unlike TABLE, the A operand of QTABLE is the operand svr\_dur of the corresponding QUEUE/DEPART pair.

1. QTABLE is simpler to user than TABLE, but
2. TABLE is more general because the A operand can be any expression whose values will be displayed in a histogram; QTABLE is used only for gathering the time elapsed when any tr proceeds between the QUEUE and DEPART pair.

QTABLE’s B, C, and D operands have the same use & meanings as those for TABLE.

Finally, the remaining blocks of SAMPLE1.GPS: RELEASE, the two DEPART blocks, and TERMINATE 1 block complete the model.

A note on facility state/status

*Notice the AVAIL. display in a .rpt file, and its value 1. At any model time,*

*if a facility is “up”, it is available to do service (AVAIL value 1), otherwise, it is “down”, or unavailable (AVAIL. value 0). IF a facility never fails, AVAIL. is always 1 during execution.[[1]](#endnote-1) <-- see Endnote*

**#5 The arrival and service distributions**

When the A and B operands of GENERATE are positive numbers, B <= A, gpss treats the {ia} distribution as Uniform with mean A and deviation (from the mean) B.

Also, A and B operands for an ADVANCE block specify a Uniform distribution with corresponding operand meanings (avg service duration and deviation-from-avg).

The GENERATE block A operand is the interarrival (abbr. “ia”) mean, and the B operand is the “half-interval” (= one half of the range of possible interarrival values = deviation). Thus, each ai is uniformly random in [10-7,10+7]. Thus, interarrival times are >= 3 and <= 17 t.u., with average value 10. Similarly, service durations specified by A and B operands of ADVANCE statement are uniformly random in [7.5-2,7.5+2].

*Other common distributions such as triangular, exponential, etc. have unequal probabilities among their values. h1 ‘HandIn results will show that different distributions (here uniform vs. exp), even if they have the same mean value.*

**#6. exp distribution values**

Reference Manual section 8.3.5 documents the many gpss built-in probability functions commonly used in mathematics/engineering/simulations. GENERATE or ADVANCE blocks need only an A operand for exp distributions, in the form: (exponential(1,0,xyz))

where the arguments of the exp call are as follows:

* a random number generator entity number; here simply specify 1; (if this argument was, instead, any legal

numeric gpss expression, its value must be an integer > 0)

* 0 denotes a horizontal shift of the distribution (0 in most 148 gpss models)
* xyz denotes the mean of the {ia} or service duration processes, respectively.

An B operand is meaningless for exp; the corresponding random variable range is (0,+∞).

**#7. HandIn requirements**

**Runs configuration rationale**

As the table below shows, 2 runs each are done for uniform and exp distribution choices.

You should expect the results for Run#1 & Run#2 to be VERY similar, but corresponding exp run results will differ in substantially in most result values.

The same seed is used in corresponding unif vs. exp runs in order to eliminate model randomness (effects of seeding) as a cause of results variance.

**Required HandIns**

1. Fill in all blank cells in the table below; each run is 5000 haircuts (Start 5000)
2. For ONLY Run#2, the first part of Report page 1, sections named: LABEL, FACILITY, and QUEUE. (Should be < 1 page; *do NOT include sections starting with the label TABLE*).
3. A copy of the source code from any run
4. Display histograms: 1) distribution of (number of cj in S) AND 2) distribution of service durations (use the Windows snipping tool, or equivalent to extract the histograms from the gpss Table Window, as already demoed in class).

**Run Results**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Run#** | **{ia} & service durations distributions** | **Seed**  format:ddddd | **UTIL. Server %busy** | **AVE.TIME avg(cj residence duration)** | **AVE.TIME**  **avg(service duration)** | **MAX**  **max(number of cj in S)** | **AVE.CONT avg(number of cj in S)** |
| 1 | **uniform** |  |  |  |  |  |  |
| 2 | **uniform** |  |  |  |  |  |  |
| 3 | **exponential** | Same seed as Run#1 |  |  |  |  |  |
| 4 | **exponential** | Same seed as Run#2 |  |  |  |  |  |

**#8 How TABLE and TABULATE work together**

All DES simulator systems have a way to create histograms. gpss calls uses a TABLE statement, and other languages call them something else (no terminology standardization ever happened across simulation languages). However, gpss has a simple and flexible way to specify generation of histogram cell entries.

In the tr labeled “demoHisto” below, a histogram is populated using two entities, a TABLE statement (technically, a gpss “statement”, not a “block”) and a TABULATE block. The code skeleton looks like:

myHisto TABLE Operands A,B,C, and D ; TABLE statement specifying a histograms’ 2-dim layout

*TABLE statements can be placed anywhere, but good practice places them with EQU, other definitions above all GENERATEs*

**:**

demoHisto GENERATE user-chosen Operands

**:**

TABULATE myHisto < -- *tabulate is a “block”; when a tr enters it, a histogram cell count increments by 1*

**:**

TERMINATE Operand

After a tr enters the TABULATE block, TABLE Operand A (it can be any legal gpss expression) is evaluated to value v, and the histogram cell where value v lies is incremented (by default) by 1.

1. Most real-world S do experience service failure scenarios. Simulating service failure (for entities that can fail) is a major topic in simulation work. CSC148 has limited time for this topic, but we will cover as much of an introduction as time permits.

   **# Example histogram – service wait duration duration for exp distributed service**

   **QTABLE use Demo**

   This histogram was NOT REQUIRED in HandIns. It demos another QTABLE statement. Recall that QTABLE does NOT need a corresponding TABULATE block as does TABLE.

   QTABLE is only capable of building histograms of wait durations for a section of the gpss source code between a QUEUE and corresponding DEPART.

   This single statement will create the desired histogram

   svrWaitDur qtable svrWait,0,3,40 ; Distribution of service wait stats

   “svrWaitDur” distinguishes several QTABLE statements in larger simulations.

   The A operand is simply the operand of the corresponding QUEUE/DEPART pair that delimits the region of the code where time spent by transactions passing through that region must be measured = > the QUEUE/DEPART must surround the (required) SEIZE block:

   queue svrWait ; Start gather service wait stats

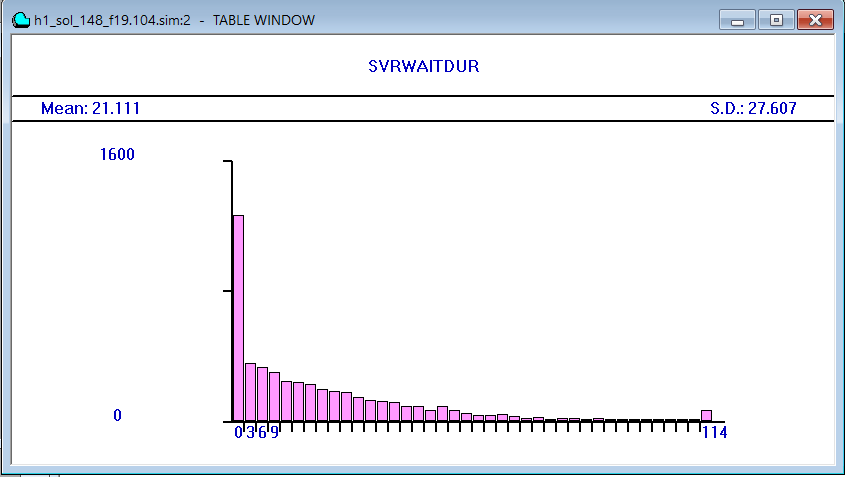
   seize barber ; Start next cj's haircut

   depart svrWait ; Finish gather service wait stats

   **The histogram display**

   The B, C, and D operands of both TABLE and QTABLE have identical meaning & use.

   Scaling the histogram horizontal axis might need some simple arithmetic so that the histogram display suits the application. Exp distributed random variables (abbreviate as rv) usually require more effort for scaling than more regular distributions such as uniform. This is because the largest values of such an rv can be very large. This display below illustrates this point – it was intentionally left as-is to illustrate what is, technically, a flawed chart. Ideally, we want the rightmost part of the horizonal axis to have cells with frequency count 0:

   Since the rv here is service wait time distributed exponentially, there were some wait times that exceeded 114.

   Given the chart here, it would be better to increase the D operand value from 40 or 50 (or even 60?) to eliminate the “hump” at scale point 114. Figuring such an issue out requires experimenting with perhaps several model runs to reach the goal.

   By the way, the service wait duration distribution is clearly not a pure exp; the bar at x=0 gives the frequency of cj that entered S but did not wait – approx. 1250 out of 5000 cj served.

   The QTABLE B operand was set to 0 so that the leftmost histogram cell height visualizes the number of cj that did not need to wait for service. Given the B and C operands, the first cell counts rv values in the range (-3,0]. [↑](#endnote-ref-1)