ANNA UNIVERSITY REGIONAL CAMPUS
COIMBATORE-641046

LABORATORY RECORD

2023-2024

NAME : ……………………………………………………

REG.NO : ……………………………………………………

BRANCH : ……………………………………………………

SUBJECT CODE : ……………………………………………………

SUBJECT TITLE : ……………………………………………………

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ANNA UNIVERSITY REGIONAL CAMPUS

COIMBATORE- 641 046.

ANNA UNIVERSITY REGIONAL CAMPUS COIMBATORE-641046

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

Certified that this is the bonafide record of practical done in **CCS357 OPTIMIZATION TECHNIQUE LABORATORY** by……………………………………Reg.No……………………...
in Third Year / Sixth Semester during 2023 – 2024 .

Staff in Charge **Head of the Department**

University Register No: ……………………………………………………………….

Submitted for the University Practical Examination held on ……………………….

Internal Examiner **External Examiner**

INDEX

EX NO	DATE	TITLE	PAGE NO.	MARKS	SIGN

EX NO	DATE	TITLE	PAGE NO.	MARKS	SIGN

Ex.No:1	SOLVING SIMPLEX MAXIMIZATION PROBLEMS USING R PROGRAMMING

AIM :

To solve Simplex Maximization Problems using R programming.

ALGORITHM:

1.Set up the coefficients for the objective function.

2.Construct a matrix to represent the coefficients of the constraints.

3.Specify the direction of each constraint by creating a vector containing the directions.

4.Define the values on the right-hand side of the constraints.

5.Use the lp function to solve the linear program.

PROGRAM:

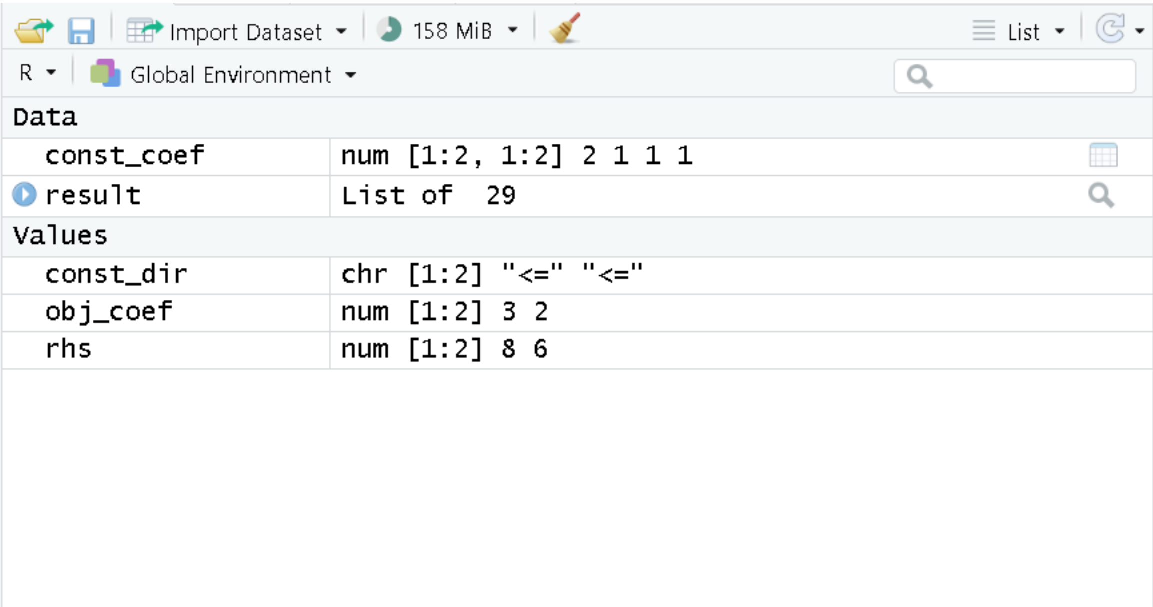
```
obj_coef<-c(3,2)

const_coef<-matrix(c(2,1,1,1),ncol=2,byrow=True)

const_dir<-c("<=","<=")

rhs<-c(8,6)

result<-lp("max",obj_coef,const_coef,const_dir,rhs)

print(result$solution)
```

OUTPUT:

Console   Terminal ×   Background Jobs ×

R 4.4.0 · ~/

```
> library(lpSolve)
> obj_coef<-c(3,2)
> const_coef<-matrix(c(2,1,1,1),ncol=2,byrow="True")
> const_dir<-c("<=","<=")
> rhs<-c(8,6)
> result<-lp("max",obj_coef,const_coef,const_dir,rhs)
> print(result$solution)
[1] 2 4
>
```



RESULT:

Thus a R program is developed to solve simplex maximization problems.

Ex.No:2	SOLVING SIMPLEX MINIMIZATION PROBLEMS USING R PROGRAMMING.

AIM :

To solve Simplex Minimization Problems using R programming.

ALGORITHM:

1.Set up the coefficients for the objective function.

2.Construct a matrix to represent the coefficients of the constraints.

3.Specify the direction of each constraint by creating a vector containing the directions.

4.Define the values on the right-hand side of the constraints.

5.Use the lp function to solve the linear program.

PROGRAM:

```
obj_coef<-c(2,3)
const_coef<-matrix(c(1,1,2,3),ncol=2,byrow=TRUE)
const_dir<-c("<=","<=")
rhs<-c(4,9)
result<-lp("min",-obj_coef,const_coef,const_dir,rhs)
print(result$solution)
```

OUTPUT:

Console | Terminal | Background Jobs

R 4.4.0 · ~/

```
> obj_coef<-c(2,3)
> const_coef<-matrix(c(1,1,2,3),ncol=2,byrow=TRUE)
> const_dir<-c("<=","<=")
> rhs<-c(4,9)
> result<-lp("min",-obj_coef,const_coef,const_dir,rhs)
> print(result$solution)
[1] 0 3
>
```

Environment | History | Connections | Tutorial

Import Dataset | 158 MiB

R | Global Environment

Data	
const_coef	num [1:2, 1:2] 1 2 1 3
result	List of 29
Values	
const_dir	chr [1:2] "<=" "<="
obj_coef	num [1:2] 2 3
rhs	num [1:2] 4 9

RESULT:

Thus a R program is developed to solve simplex minimization problems.

Ex.No:3	SOLVING MIXED CONSTRAINTS PROBLEMS – BIG M & TWO PHASE METHOD USING TORA.

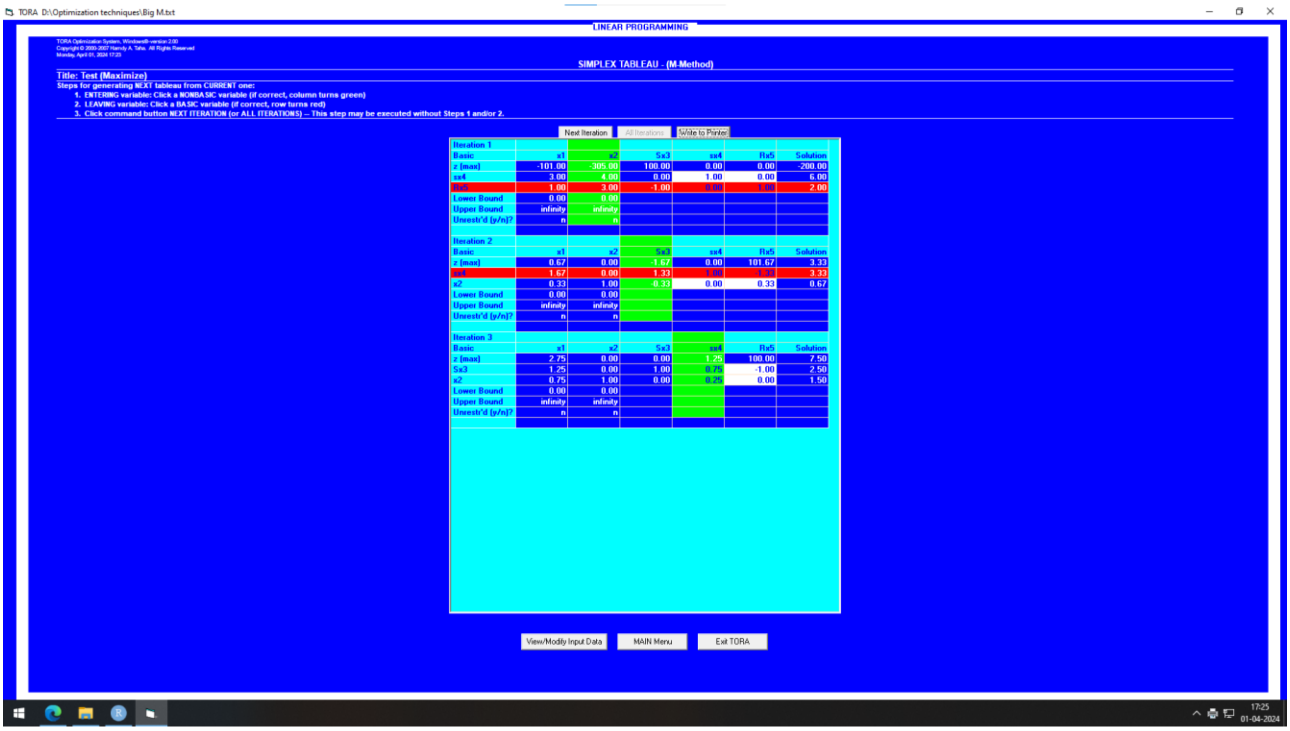
AIM :

To solve mixed constraints problems – Big M & Two Phase method using TORA.

PROCEDURE:

1. Define the linear programming problem, including the objective function and constraints.
2. Introduce slack variables for mixed constraints, setting them with a large positive value (M).
3. Phase one aims to find an initial feasible solution by minimizing the sum of artificial variables. Phase two optimizes the original problem using the simplex method.
4. Solve the original linear programming problem using the simplex method, starting from the initial feasible solution obtained in phase one.
5. Extract the optimal solution, including decision variable values, and conduct sensitivity analysis to assess solution robustness.

OUTPUT:



RESULT:

Thus mixed constraint problems – Big M & Two Phase method are solved using TORA.

Ex.No:4	SOLVING TRANSPORTATION PROBLEMS USING R.

AIM :

To solve Transportation Problems using R programming.

ALGORITHM:

1.Install and load the "transport" package.

2.Specify supply, demand, and transportation costs.

3.Utilize the transport function to find the optimal transportation plan.

4.Store the solution and associated information in a variable.

5.Print the optimized transportation plan and total cost.

PROGRAM:

```
install.packages("transport")
library(transport)
supply<-c(100,150,200)
demand<-c(120,180,150)
costs<-matrix(c(4,6,8,9,5,7,3,2,9),nrow=3,byrow=TRUE)
res<-transport(supply,demand,costs)
print(res)
```

OUTPUT:

Console | Terminal | Background Jobs

R 4.4.0 · ~/

```
> library(transport)
> supply<-c(100,150,200)
> demand<-c(120,180,150)
> costs<-matrix(c(4,6,8,9,5,7,3,2,9),nrow=3,byrow=TRUE)
> res<-transport(supply,demand,costs)
> print(res)
  from to mass
1    1  1  100
2    2  3  150
3    3  1   20
4    3  2  180
>
```

Data	
const_coef	num [1:2, 1:2] 1 2 1 3
costs	num [1:3, 1:3] 4 9 3 6 5 2 8 7 9
res	4 obs. of 3 variables
result	List of 29
Values	
const_dir	chr [1:2] "<=" "<="
demand	num [1:3] 120 180 150
obj_coef	num [1:2] 2 3
rhs	num [1:2] 4 9
supply	num [1:3] 100 150 200

RESULT:

Thus R Program for solving transportation problems is developed.

Ex.No:5	SOLVING ASSIGNMENT PROBLEMS USING R.

AIM :

To solve Assignment Problems using R programming.

ALGORITHM:

1.Install and load the "lpSolve" package for linear programming.

2.Define the cost matrix representing the assignment costs between agents and tasks.

3.Create an LP model with the objective to minimize the total assignment cost. Define constraints ensuring that each agent is assigned exactly one task, and each task is assigned to exactly one agent.

4.Use the solve function to find the optimal solution to the LP model.

5. Print the assignment matrix showing which agent is assigned to which task, and calculate the total cost of the assignment.

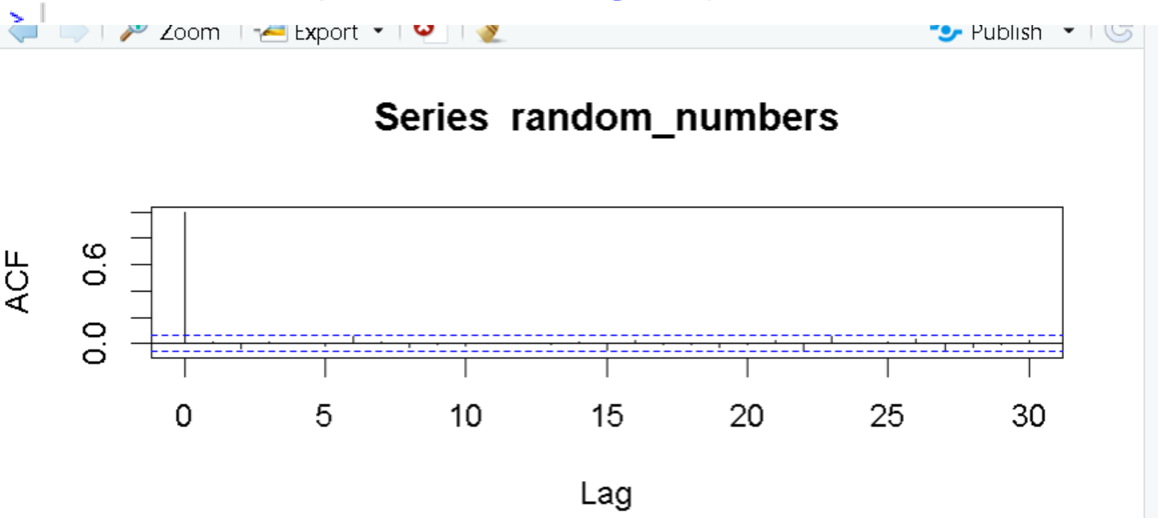
PROGRAM:

```
install.packages("lpSolve")

library(lpSolve)

cost_matrix <- matrix(c(

  10, 7, 3, 5,

  8, 6, 9, 4,

  7, 9, 6, 2,

  2, 4, 8, 7

), nrow = 4, byrow = TRUE)

num_agents <- nrow(cost_matrix)

num_tasks <- ncol(cost_matrix)

assignment_lp <- lp(direction = "min",
```

```
                objective.in = as.vector(cost_matrix),
                const.mat = rbind(matrix(1, nrow = num_agents),
                              t(matrix(1, nrow = num_tasks))),
                const.dir = c("==", "=="),
                const.rhs = c(1, 1),
                all.bin = TRUE)  # Binary variables
solution <- solve(assignment_lp)
assignment <- matrix(solution$solution, nrow = num_agents, ncol = num_tasks,
byrow = TRUE)
print("Assignment matrix:")
print(assignment)
total_cost <- sum(cost_matrix * assignment)
print(paste("Total cost:", total_cost))
```

OUTPUT:

```
> assignment <- matrix(assignment_lp$solution, nrow = num_agents, ncol = num_tas
s, byrow = TRUE)
> print("Assignment matrix:")
[1] "Assignment matrix:"
> print(assignment)
     [,1] [,2] [,3] [,4]
[1,]    0    0    0    1
[2,]    0    0    0    0
[3,]    0    0    0    0
[4,]    0    0    0    0
> total_cost <- sum(cost_matrix * assignment)
>
```



RESULT:

Thus R Program for solving assignment problems is developed.

Ex.No:6	SOLVING OPTIMIZATION PROBLEMS USING LINGO

AIM :

To solve Optimization Problems using LINGO.

ALGORITHM:

1.Define the objective function and constraints. The objective is to minimize the expression given, subject to the constraints provided.

2.Declare decision variables and specify their binary nature using the @BIN directive.

3.Set up the optimization model by specifying the objective function, constraints, and variable types.

4.Use LINGO to solve the optimization problem and find the optimal solution.

5.Interpret the results, including the optimal values of decision variables and the minimized objective function value, to derive actionable insights.

PROGRAM:

```
MIN = 22*x11 + 28*x12 + 30*x13 + 18*x14 + 18*x21 + 0*x22 + 27*x23 + 22*x24
+ 26*x31 + 20*x32 + 28*x33 + 28*x34 + 16*x41 + 22*x42 + 0*x43+ 14*x44 +
21*x51 + 0*x52 + 25*x53 + 28*x54;

x11 + x12 + x13 + x14 <= 1;

x21 + x23 + x24 <= 1;

x31 + x32 + x33 + x34 <= 1;

x41 + x42 + x44 <= 1;

x51 + x53 + x54 <= 1;

x11 + x21 + x31 + x41 + x51 = 1;

x12 + x32 + x42 = 1;

x13 + x23 + x33 + x53 = 1;

x14 + x24 + x34 + x44 + x54 = 1;
```

@BIN(x12);

@BIN(x13);

@BIN(x14);

@BIN(x21);

@BIN(x23);

@BIN(x24);

@BIN(x31);

@BIN(x32);

@BIN(x33);

@BIN(x34);

@BIN(x41);

@BIN(x42);

@BIN(x44);

@BIN(x51);

@BIN(x53);

@BIN(x54);

OUTPUT:

File Edit Solver Window Help

```
LINGO/WIN64 20.0.23 (5 Sep 2023 ), LINDO API 14.0.5099.295

Licensee info: Eval Use Only
License expires: 27 OCT 2024

Global optimal solution found.
Objective value:                              77.00000
Objective bound:                              77.00000
Infeasibilities:                              0.000000
Extended solver steps:                               0
Total solver iterations:                             0
Elapsed runtime seconds:                          0.09

Model Class:                                      MILP

Total variables:                    20
Nonlinear variables:                 0
Integer variables:                  17

Total constraints:                  10
Nonlinear constraints:               0

Total nonzeros:                     51
Nonlinear nonzeros:                  0


                              Variable           Value        Reduced Cost
                                   X11        0.000000            22.00000
                                   X12        0.000000            28.00000
                                   X13        0.000000            30.00000
                                   X14        0.000000            18.00000
                                   X21        1.000000            18.00000
                                   X22        0.000000            0.000000
                                   X23        0.000000            27.00000
                                   X24        0.000000            22.00000
                                   X31        0.000000            26.00000
```

File Edit Solver Window Help

Variable	Value	Reduced Cost
X11	0.000000	22.00000
X12	0.000000	28.00000
X13	0.000000	30.00000
X14	0.000000	18.00000
X21	1.000000	18.00000
X22	0.000000	0.000000
X23	0.000000	27.00000
X24	0.000000	22.00000
X31	0.000000	26.00000
X32	1.000000	20.00000
X33	0.000000	28.00000
X34	0.000000	28.00000
X41	0.000000	16.00000
X42	0.000000	22.00000
X43	0.000000	0.000000
X44	1.000000	14.00000
X51	0.000000	21.00000
X52	0.000000	0.000000
X53	1.000000	25.00000
X54	0.000000	28.00000

Row	Slack or Surplus	Dual Price
1	77.00000	-1.000000
2	1.000000	0.000000
3	0.000000	0.000000
4	0.000000	0.000000
5	0.000000	0.000000
6	0.000000	0.000000
7	0.000000	0.000000
8	0.000000	0.000000
9	0.000000	0.000000
10	0.000000	0.000000

RESULT:

Thus an optimization problem is solved using LINGO.

Ex.No:7	STUDYING PRIMAL-DUAL RELATIONSHIPS IN LP USING TORA.

AIM :

To study Primal Dual relationships in LP using TORA.

PROCEDURE:

1.Studying primal-dual relationships using the Temporally Ordered Routing Algorithm (TORA) requires a different approach than traditional optimization methods.

2.Recognize that TORA routing decisions can be viewed as solutions to an optimization problem analogous to linear programming. Understand how TORA optimizes routing decisions based on network conditions and constraints.

3.Define LP optimization objectives relevant to TORA routing that align with primal and dual objectives in LP problems. For instance, minimizing total energy consumption, maximizing network throughput, or minimizing packet transmission delay can be considered LP optimization.

4.Map the identified LP optimization objectives to a primal LP problem formulation. Define decision variables, objective function, and constraints that represent the routing decisions and network parameters optimized by TORA.

5. Derive the dual LP problem from the primal LP problem by introducing dual variables associated with the primal problem's constraints. The dual problem should offer insight into the trade-offs and relationships between different optimization objectives in TORA routing.

6.Use LP solvers or optimization software to simulate TORA routing scenarios as LP problems. Convert TORA routing decisions and network parameters into LP problem formulations and input them into LP solvers for analysis.

7.Define LP performance metrics that capture the optimization objectives and primal-dual relationships in TORA routing. Assess how well TORA routing decisions align with the primal and dual optimization objectives based on LP solver outputs.

8.Analyze the LP simulation results to understand how TORA routing decisions relate to the primal and dual LP objectives. Investigate how changes in network conditions or optimization parameters impact the primal and dual solutions derived from LP simulations.

9.Draw conclusions based on the analysis of primal-dual relationships in TORA routing as LP problems. Identify trade-offs, synergies, or conflicts between different optimization objectives and assess the effectiveness of TORA in achieving optimal or near-optimal solutions.

10.Iterate on the study by refining LP simulation parameters, adjusting optimization objectives, or exploring alternative LP formulations.Continuously refine your understanding of primal-dual relationships in TORA routing as LP problems to deepen insights into network optimization.

RESULT:

Thus primal dual relationships in LP is studied using TORA.

Ex.No:8	SOLVING LP PROBLEMS USING DUAL SIMPLEX METHOD USING TORA

AIM :

To solve LP Problems using Dual Simplex method using TORA.

PROCEDURE:

1.Define the LP problem, including the objective function and constraints.

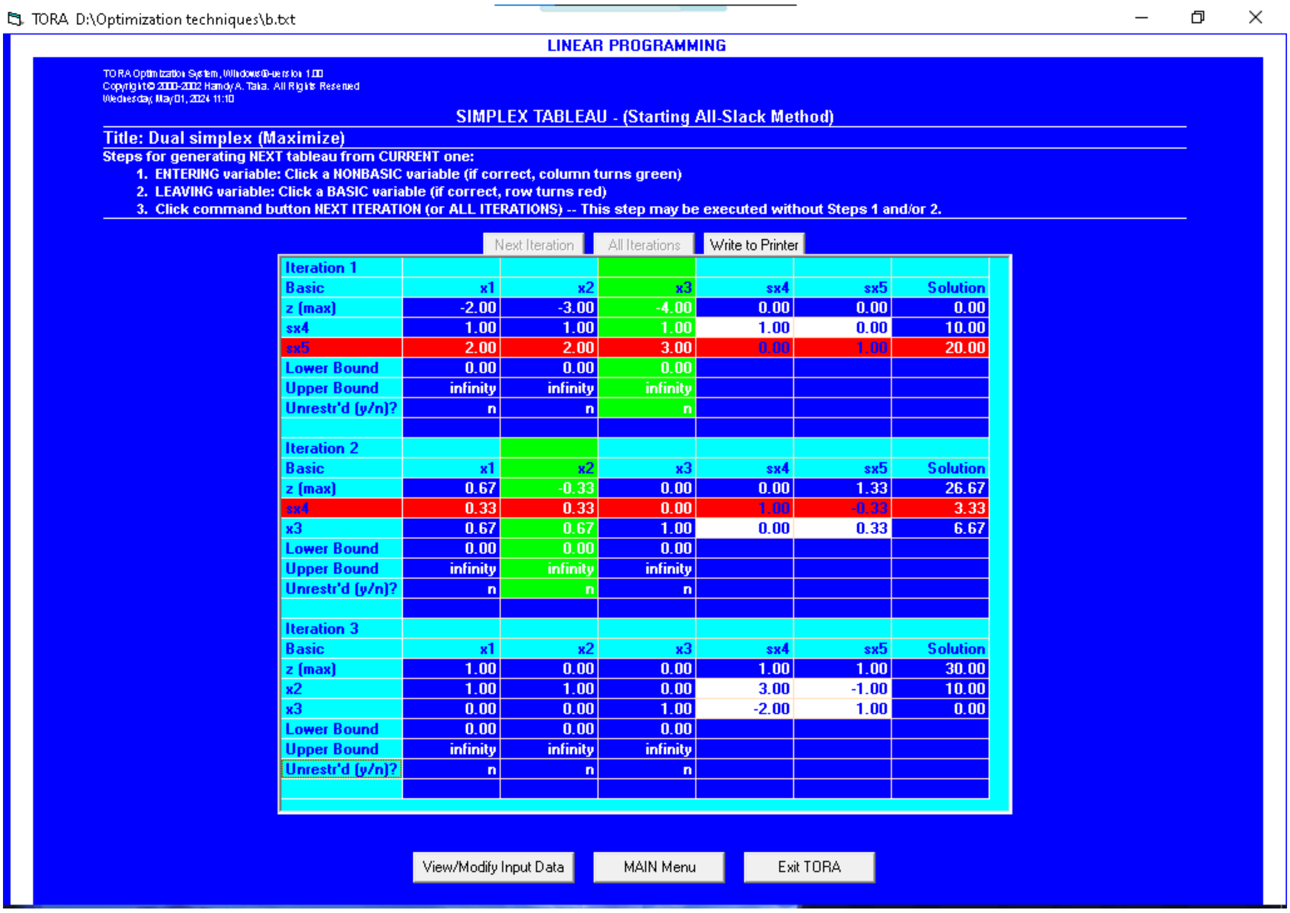
2.Input the LP problem into TORA.

3.Choose the Dual Simplex Method in TORA.

4.Let TORA execute the Dual Simplex Method.

5.Review TORA's output for the optimal solution and analysis.

OUTPUT:


TORA D:\Optimization techniques\b.txt

LINEAR PROGRAMMING

TORA Optimization System, Windows®-version 1.00
Copyright © 2000-2002 Hamdy A. Taha. All Rights Reserved
Wednesday, May 01, 2024 11:10

SIMPLEX TABLEAU - (Starting All-Slack Method)

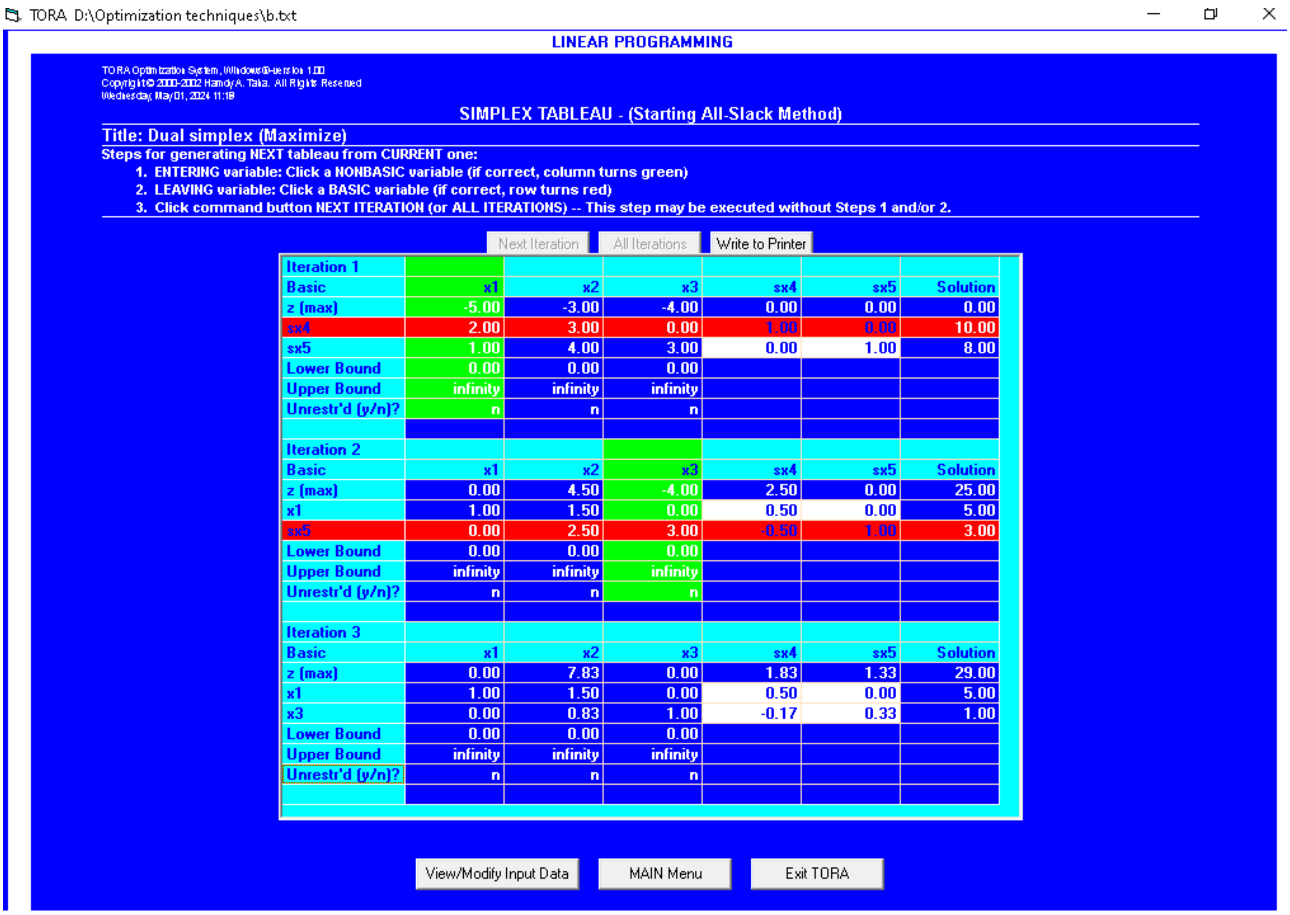
Title: Dual simplex (Maximize)

Steps for generating NEXT tableau from CURRENT one:
1. ENTERING variable: Click a NONBASIC variable (if correct, column turns green)
2. LEAVING variable: Click a BASIC variable (if correct, row turns red)
3. Click command button NEXT ITERATION (or ALL ITERATIONS) -- This step may be executed without Steps 1 and/or 2.

Next Iteration | All Iterations | Write to Printer

Iteration 1						
Basic	x1	x2	x3	sx4	sx5	Solution
z (max)	-2.00	-3.00	-4.00	0.00	0.00	0.00
sx4	1.00	1.00	1.00	1.00	0.00	10.00
sx5	2.00	2.00	3.00	0.00	1.00	20.00
Lower Bound	0.00	0.00	0.00			
Upper Bound	infinity	infinity	infinity			
Unrestr'd (y/n)?	n	n	n			
Iteration 2						
Basic	x1	x2	x3	sx4	sx5	Solution
z (max)	0.67	-0.33	0.00	0.00	1.33	26.67
sx4	0.33	0.33	0.00	1.00	-0.33	3.33
x3	0.67	0.67	1.00	0.00	0.33	6.67
Lower Bound	0.00	0.00	0.00			
Upper Bound	infinity	infinity	infinity			
Unrestr'd (y/n)?	n	n	n			
Iteration 3						
Basic	x1	x2	x3	sx4	sx5	Solution
z (max)	1.00	0.00	0.00	1.00	1.00	30.00
x2	1.00	1.00	0.00	3.00	-1.00	10.00
x3	0.00	0.00	1.00	-2.00	1.00	0.00
Lower Bound	0.00	0.00	0.00			
Upper Bound	infinity	infinity	infinity			
Unrestr'd (y/n)?	n	n	n			

View/Modify Input Data | MAIN Menu | Exit TORA



Iteration 1						
Basic	x1	x2	x3	sx4	sx5	Solution
z (max)	-5.00	-3.00	-4.00	0.00	0.00	0.00
sx4	2.00	3.00	0.00	1.00	0.00	10.00
sx5	1.00	4.00	3.00	0.00	1.00	8.00
Lower Bound	0.00	0.00	0.00			
Upper Bound	infinity	infinity	infinity			
Unrestr'd (y/n)?	n	n	n			
Iteration 2						
Basic	x1	x2	x3	sx4	sx5	Solution
z (max)	0.00	4.50	-4.00	2.50	0.00	25.00
x1	1.00	1.50	0.00	0.50	0.00	5.00
sx5	0.00	2.50	3.00	-0.50	1.00	3.00
Lower Bound	0.00	0.00	0.00			
Upper Bound	infinity	infinity	infinity			
Unrestr'd (y/n)?	n	n	n			
Iteration 3						
Basic	x1	x2	x3	sx4	sx5	Solution
z (max)	0.00	7.83	0.00	1.83	1.33	29.00
x1	1.00	1.50	0.00	0.50	0.00	5.00
x3	0.00	0.83	1.00	-0.17	0.33	1.00
Lower Bound	0.00	0.00	0.00			
Upper Bound	infinity	infinity	infinity			
Unrestr'd (y/n)?	n	n	n			

RESULT:

Thus LP Problems are solved using Dual Simplex Method in TORA.

Ex.No:9	SENSITIVITY & POST OPTIMALITY ANALYSIS USING LINGO.

AIM :

To study sensitivity and post optimality analysis using LINGO.

PROCEDURE:

1.Sensitivity Analysis:

•Modify the coefficients of the objective function to observe how changes in these coefficients affect the optimal objective value. You can increase or decrease the coefficients to see if the optimal solution remains unchanged or if there's a change in the objective value.

•Change the RHS values of the constraints to understand how variations in resource availability or demand affect the optimal solution. Increase or decrease the RHS values within feasible ranges and observe the impact on the optimal solution.

•LINGO provides shadow prices (dual values) associated with each constraint in the LP model. Analyze the shadow prices to understand the impact of relaxing or tightening constraints on the optimal solution. Positive shadow prices indicate that increasing the RHS values of the corresponding constraints will lead to an increase in the objective value, and vice versa.

•Determine the allowable increase and decrease in objective function coefficients or RHS values before the optimal solution changes. This helps identify the range within which the optimal solution remains unchanged.

2.Post-Optimality Analysis:

•Analyze the reduced costs associated with decision variables to identify variables that are candidates for additional investment or reduction. Negative reduced costs indicate that increasing the variable's value could improve the objective value.

•LINGO generates sensitivity reports that provide detailed information about the sensitivity of the optimal solution to changes in problem parameters. Review these reports to understand the impact of parameter variations on the optimal solution and make informed decisions.

•Conduct feasibility analysis to ensure that the optimal solution remains feasible even when parameters change. Verify that all constraints are satisfied within acceptable tolerance levels.

•Explore different scenarios by systematically varying problem parameters and observing changes in the optimal solution. This helps in understanding the robustness of the solution and identifying potential risks or opportunities.

3.Interpretation and Decision-Making:

•Interpret the results of sensitivity and post-optimality analysis to gain insights into the behavior of the LP model under different conditions.

•Use the analysis findings to make informed decisions regarding resource allocation, capacity planning, pricing strategies, and other aspects of the optimization problem.

•Communicate the results and recommendations to stakeholders effectively, highlighting key insights and implications for decision-making.

RESULT:

Thus sensitivity and post optimality analysis is studied using LINGO.

Ex.No:10	SOLVING SHORTEST ROUTE PROBLEMS USING OPTIMIZATION SOFTWARE

AIM :

To solve shortest route problems using optimization software-TORA.

PROCEDURE:

1.Define the shortest route problem.

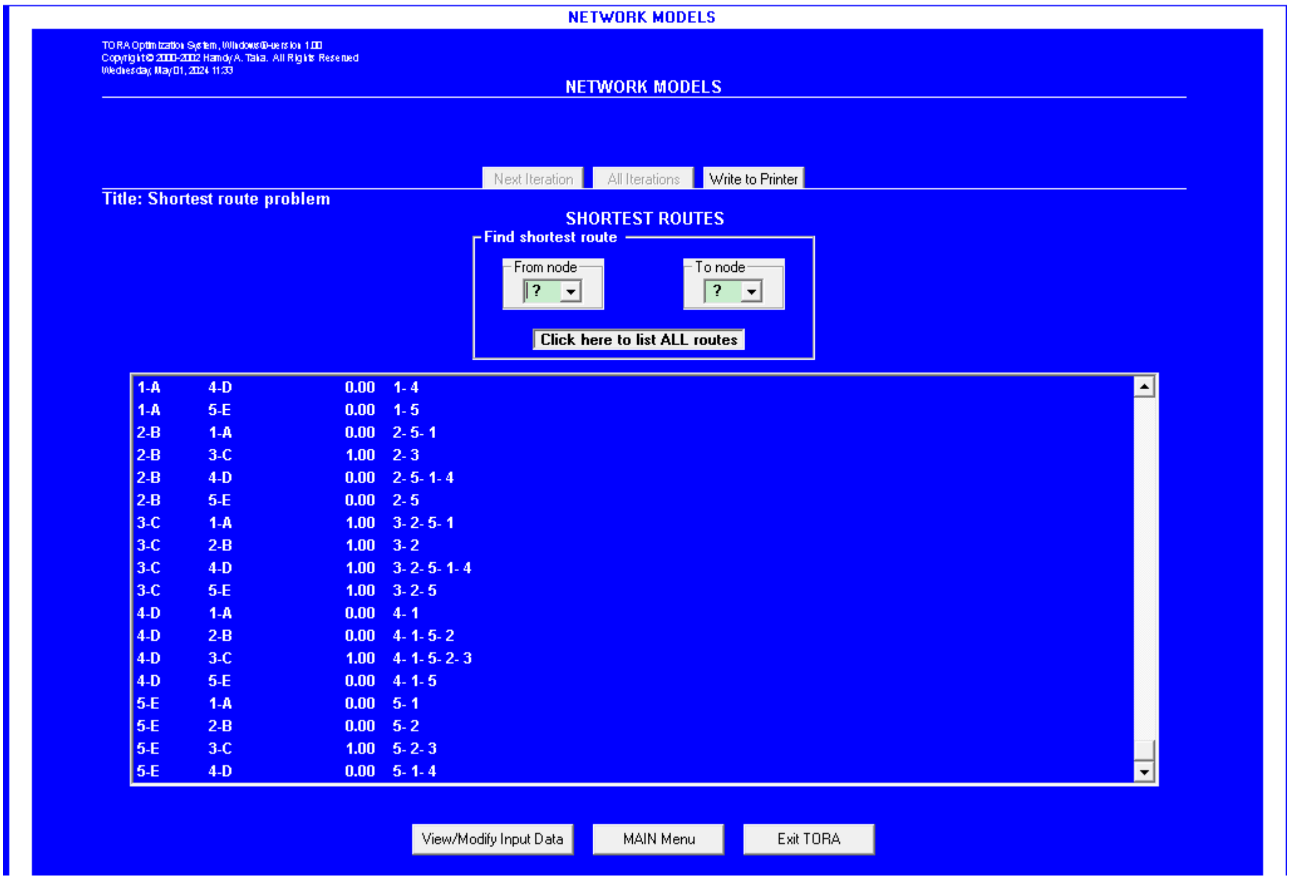
2.Input the problem into TORA.

3.Select TORA's shortest path algorithm.

4.Let TORA find the shortest route.

5.Check TORA's output for the optimal route.

OUTPUT:



1-A	4-D	0.00	1- 4
1-A	5-E	0.00	1- 5
2-B	1-A	0.00	2- 5- 1
2-B	3-C	1.00	2- 3
2-B	4-D	0.00	2- 5- 1- 4
2-B	5-E	0.00	2- 5
3-C	1-A	1.00	3- 2- 5- 1
3-C	2-B	1.00	3- 2
3-C	4-D	1.00	3- 2- 5- 1- 4
3-C	5-E	1.00	3- 2- 5
4-D	1-A	0.00	4- 1
4-D	2-B	0.00	4- 1- 5- 2
4-D	3-C	1.00	4- 1- 5- 2- 3
4-D	5-E	0.00	4- 1- 5
5-E	1-A	0.00	5- 1
5-E	2-B	0.00	5- 2
5-E	3-C	1.00	5- 2- 3
5-E	4-D	0.00	5- 1- 4

RESULT:

Thus shortest route problem is solved using TORA software.

Ex.No:11	**SOLVING PROJECT MANAGEMENT PROBLEMS USING OPTIMIZATION SOFTWARE**

AIM :

To solve project management problems using optimization software – TORA.

PROCEDURE:

1.Define the project management problem, including tasks, durations, and dependencies.

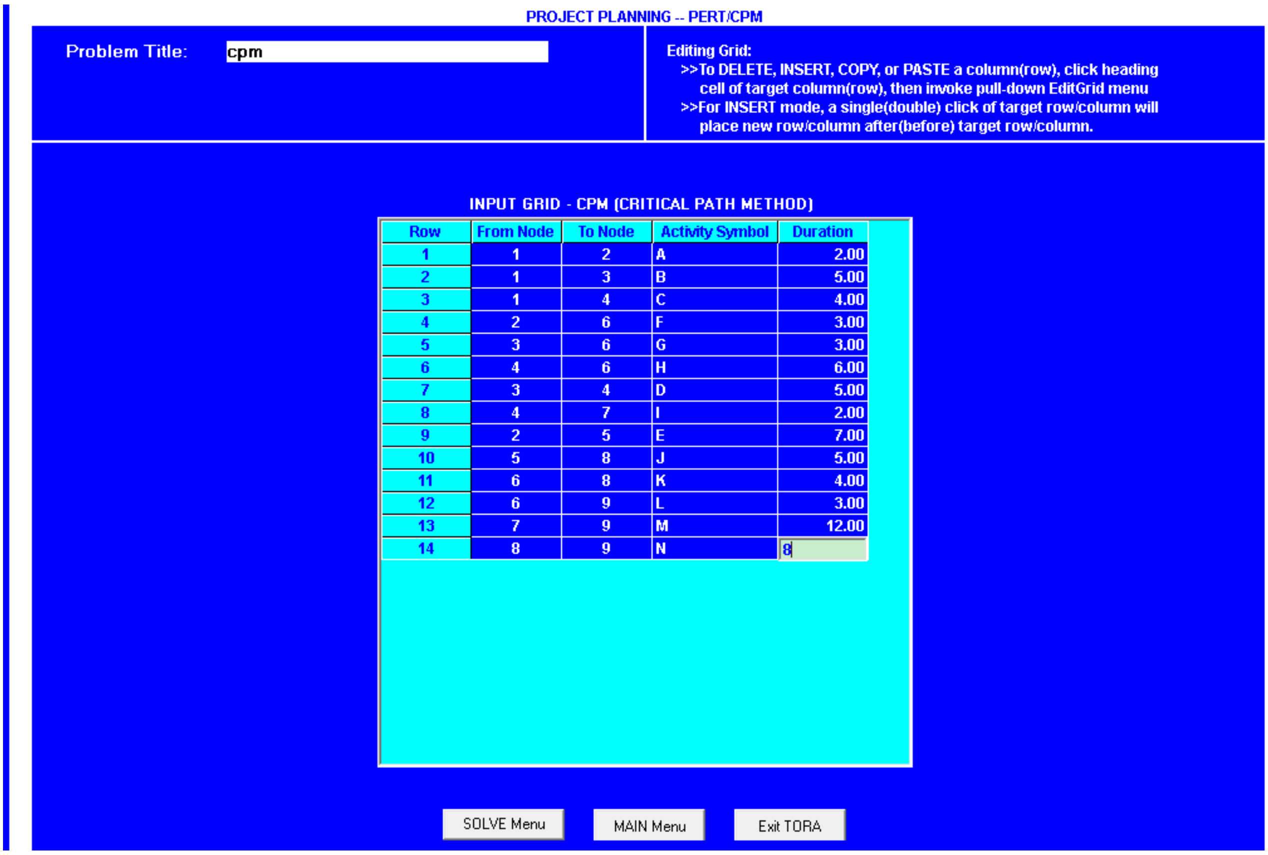
2.Input the project management problem into TORA.

3.Choose TORA's project scheduling algorithm.(CPM/PERT)

4.Allow TORA to optimize the project schedule.

5.Review TORA's output for the optimized project schedule.

OUTPUT :



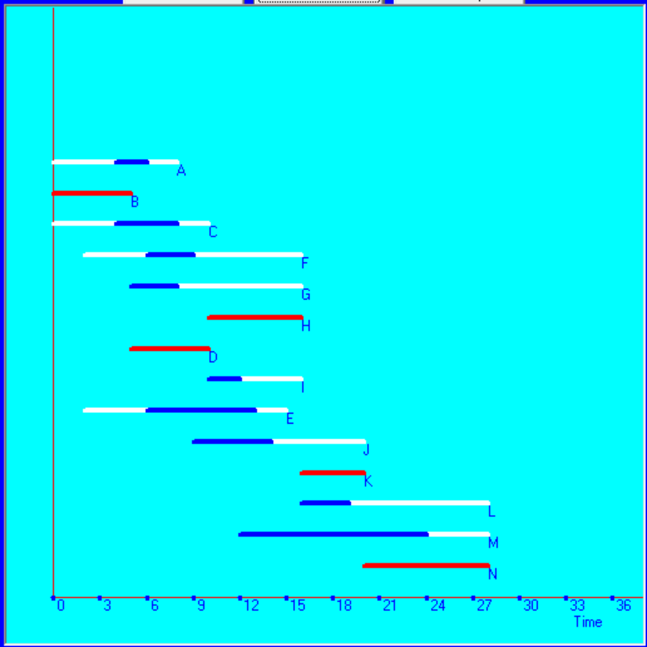
Row	From Node	To Node	Activity Symbol	Duration
1	1	2	A	2.00
2	1	3	B	5.00
3	1	4	C	4.00
4	2	6	F	3.00
5	3	6	G	3.00
6	4	6	H	6.00
7	3	4	D	5.00
8	4	7	I	2.00
9	2	5	E	7.00
10	5	8	J	5.00
11	6	8	K	4.00
12	6	9	L	3.00
13	7	9	M	12.00
14	8	9	N	8

PROJECT PLANNING -- PERT/CPM

CRITICAL PATH METHOD SCHEDULE

Zoom In | Zoom Out | Print Graph

	cpm		
Activity	Duration	Earliest Start	La
1-2: A	2.00	0.00	
1-3: B	5.00	0.00	
1-4: C	4.00	0.00	
2-6: F	3.00	2.00	
3-6: G	3.00	5.00	
4-6: H	6.00	10.00	
3-4: D	5.00	5.00	
4-7: I	2.00	10.00	
2-5: E	7.00	2.00	
5-8: J	5.00	9.00	
6-8: K	4.00	16.00	
6-9: L	3.00	16.00	
7-9: M	12.00	12.00	



Experiment with schedule changes

Maximum change is limited to Total Float

Change start of A by

ACTIVITY A: START TIME = 4, DELAY relative to its EARLIEST START = 4, FREE FLOAT = 0. Because DELAY exceeds FREE FLOAT by 4, succeeding activities {F, E} cannot start any earlier than time 6.

CAUTION: Activities following {F, E}, if any, are not checked for possible delay. This step is done manually by assigning zero delay to {F, E}.

View/Modify Input Data | MAIN Menu | Exit TORA

PROJECT PLANNING -- PERT/CPM

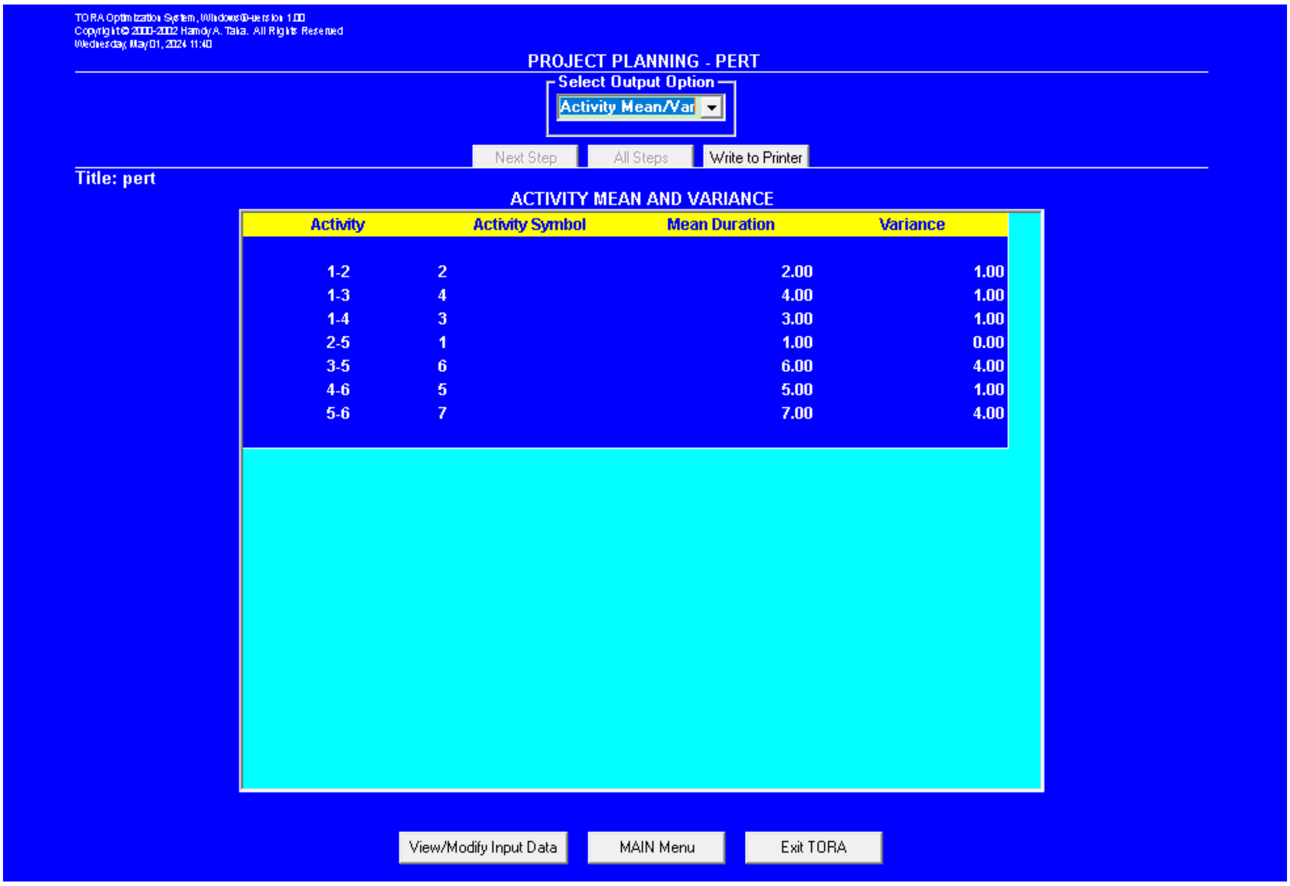
Problem Title: pert

Editing Grid:
>>To DELETE, INSERT, COPY, or PASTE a column(row), click heading cell of target column(row), then invoke pull-down EditGrid menu
>>For INSERT mode, a single(double) click of target row/column will place new row/column after(before) target row/column.

INPUT GRID - PERT (PROGRAM EVALUATION & REVIEW TECHNIQUE)

	From Node	To Node	Activity Symbol	a	m	b
1	1	2	2	1.00	1.00	7.00
2	1	3	4	1.00	4.00	7.00
3	1	4	3	2.00	2.00	8.00
4	2	5	1	1.00	1.00	1.00
5	3	5	6	2.00	5.00	14.00
6	4	6	5	2.00	5.00	8.00
7	5	6	7	3.00	6.00	15

SOLVE Menu | MAIN Menu | Exit TORA



Activity	Activity Symbol	Mean Duration	Variance
1-2	2	2.00	1.00
1-3	4	4.00	1.00
1-4	3	3.00	1.00
2-5	1	1.00	0.00
3-5	6	6.00	4.00
4-6	5	5.00	1.00
5-6	7	7.00	4.00

RESULT:

Thus, project management problems are solved using TORA.

Ex.No:12	TESTING RANDOM NUMBERS AND RANDOM VARIATES FOR THEIR UNIFORMITY

AIM :

To test random numbers and random variates for their uniformity.

ALGORITHM :

1.Generate a vector of 1000 random numbers using the uniform distribution.

2.Plot a histogram of the random numbers with 20 breaks using hist().

3.Perform a chi-square test on the binned data to test for uniformity using chisq.test().

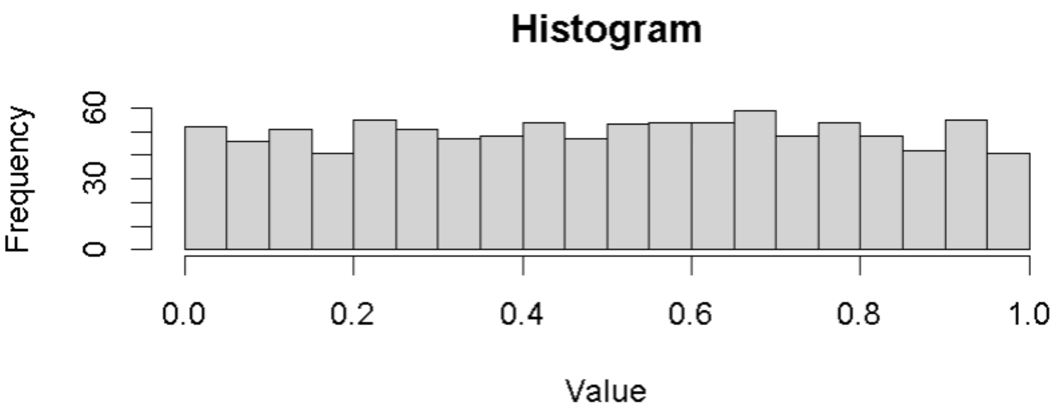
4.Perform a Kolmogorov-Smirnov test to compare the sample distribution with the uniform distribution using ks.test().

5.Create a quantile-quantile plot to visually assess the goodness of fit between the sample and theoretical uniform distribution using qqplot().

PROGRAM :

```
random_numbers = runif(1000)

hist(random_numbers,breaks=20,main-
"Histogram",xlab="Value",ylab="Frequency")

chi<-chisq.test(table(cut(random_numbers,breaks=20)))

test <- ks.test(random_numbers,"punif")

print(chi)

print(test)

qqplot(random_numbers, runif(1000), main = "Q-Q Plot", xlab = "Theoretical
Quantiles", ylab = "Sample Quantiles")

abline(0, 1)
```

OUTPUT :



Console Terminal Background Jobs

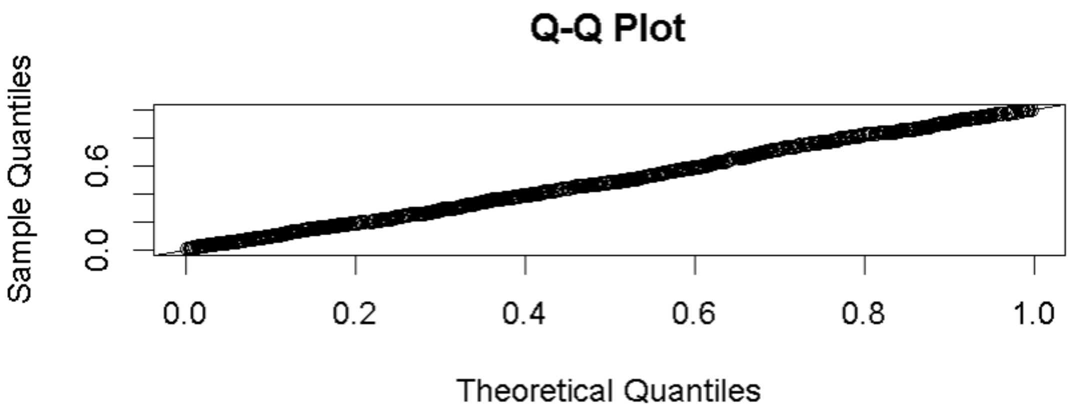
R 4.4.0 · ~/

```
> random_numbers = runif(1000)
> hist(random_numbers,breaks=20,main="Histogram",xlab="Value",ylab="Frequency")
> chi<-chisq.test(table(cut(random_numbers,breaks=20)))
> test <- ks.test(random_numbers,"punif")
> print(chi)

        Chi-squared test for given probabilities

data:  table(cut(random_numbers, breaks = 20))
X-squared = 7.24, df = 19, p-value = 0.9928

> |
```

R Global Environment

assignment_lp	List of 29
chi	List of 9
const_coef	num [1:2, 1:2] 2 1 1 1
cost_matrix	num [1:4, 1:4] 10 8 7 2 7 6 9 4 3 9 ...
costs	num [1:3, 1:3] 4 9 3 6 5 2 8 7 9
ks_test	List of 6
res	4 obs. of 3 variables
result	List of 29
test	List of 6
Values	
const_dir	chr [1:2] "<=" "<="
demand	num [1:3] 120 180 150

```
> print(test)

        Asymptotic one-sample Kolmogorov-Smirnov test

data:  random_numbers
D = 0.016381, p-value = 0.9512
alternative hypothesis: two-sided

> qqplot(random_numbers, runif(1000), main = "Q-Q Plot", xlab = "Theoretical Quant
iles", ylab = "Sample Quantiles")
> abline(0, 1)
>
```



RESULT:

Thus testing random numbers and random variates for their uniformity is done using R programming.

Ex.No:13	**TESTING RANDOM NUMBERS AND RANDOM VARIATES FOR THEIR INDEPENDENCE**

AIM :

To test random numbers and random variates for their independence.

ALGORITHM :

1.Generate 1000 random numbers from a uniform distribution.

2.Plot a histogram of the random numbers with 20 breaks.

3.Create a normal quantile-quantile plot to assess the normality of the distribution.

4.Perform a Kolmogorov-Smirnov test to assess whether the data follows a uniform distribution.

5.Perform a chi-square test to assess the goodness of fit between the observed and expected frequencies.

6.Plot the autocorrelation function (ACF) of the random numbers.

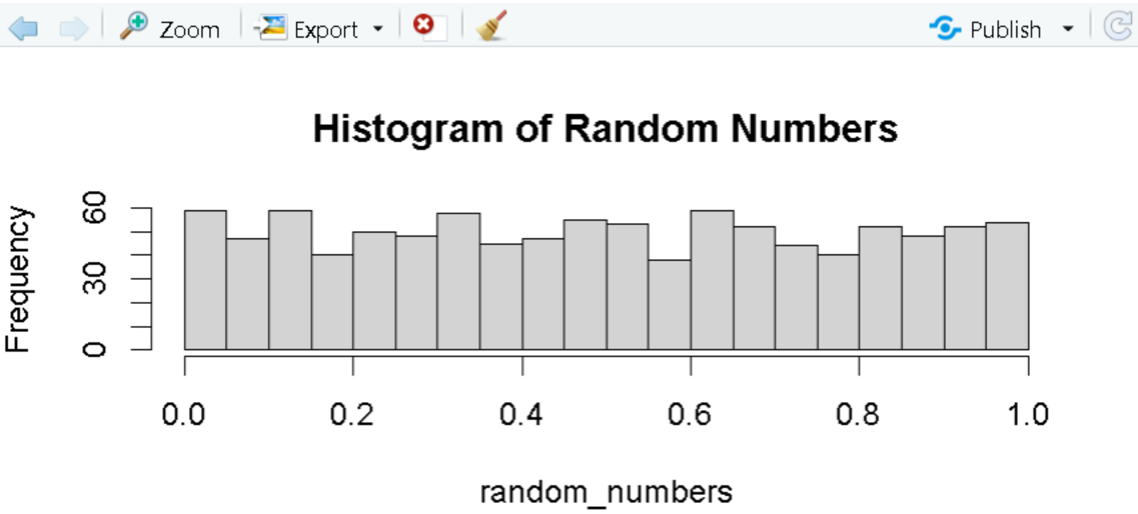
7.Display the frequency table of the random numbers.

PROGRAM:

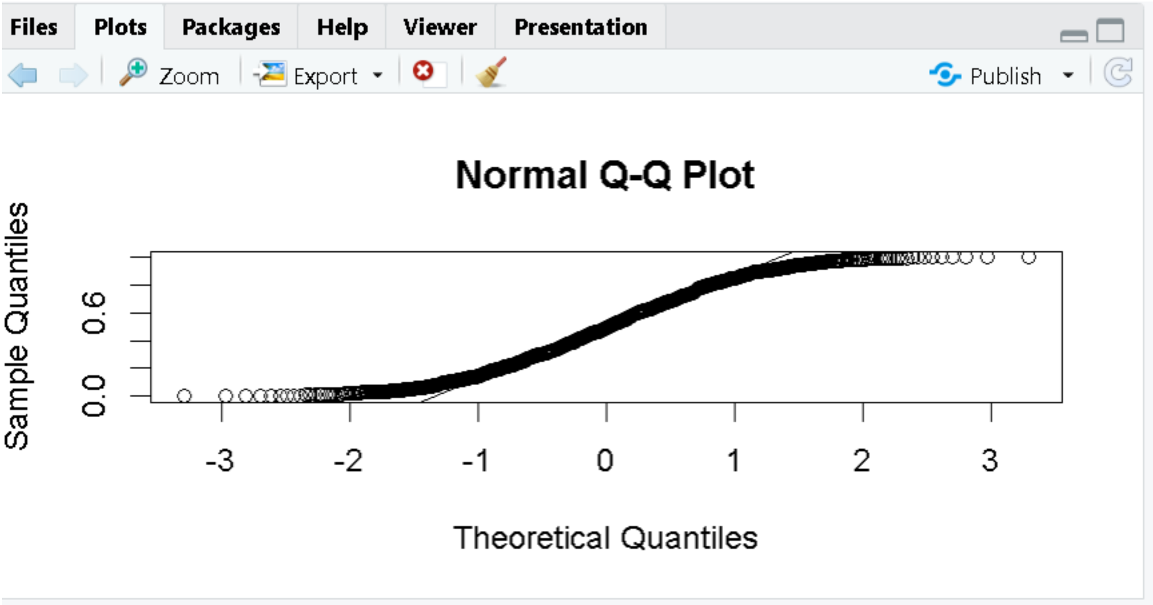
```
random_numbers <- runif(1000)

hist(random_numbers, breaks = 20, main = "Histogram of Random Numbers")

qqnorm(random_numbers)

qqline(random_numbers)

ks_test <- ks.test(random_numbers, "punif")

print(ks_test)

expected_counts <- rep(1000/20, 20)

expected_counts <- expected_counts / sum(expected_counts)

chi_square_test <- chisq.test(table(cut(random_numbers, breaks = 20)), p =
expected_counts)
```

```
print(chi_square_test)
acf(random_numbers)
frequency_table <- table(cut(random_numbers, breaks = 20))
print(frequency_table)
```

OUTPUT:

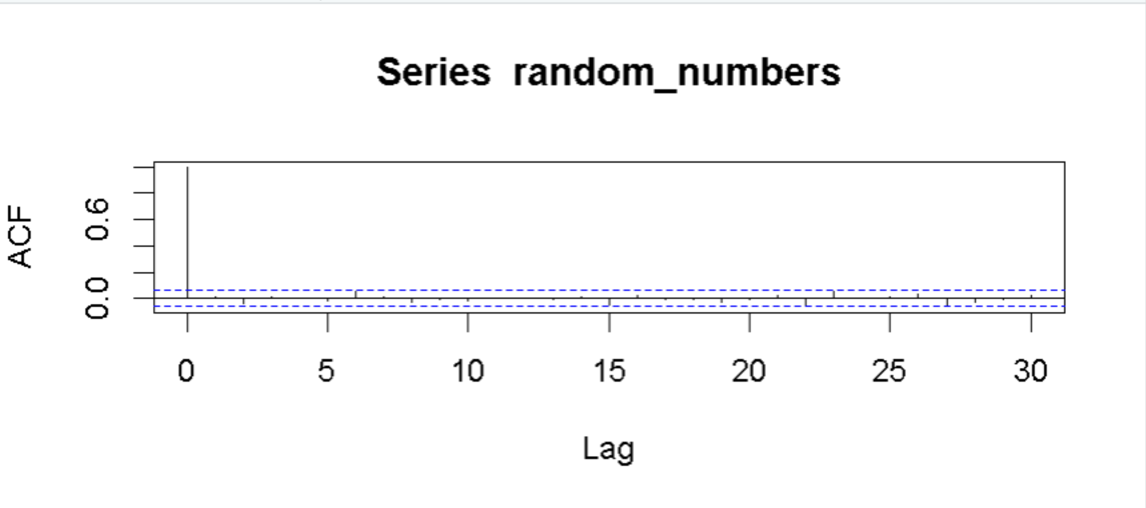


chi_square_test	List of 9
const_coef	num [1:2, 1:2] 2 1 1 1
cost_matrix	num [1:4, 1:4] 10 8 7 2 7 6 9 4 3 9 ...
costs	num [1:3, 1:3] 4 9 3 6 5 2 8 7 9
ks_test	List of 6
res	4 obs. of 3 variables
result	List of 29
test	List of 6
Values	
const_dir	chr [1:2] "<=" "<="
demand	num [1:3] 120 180 150




```
> qqnorm(random_numbers)
> qqline(random_numbers)
> ks_test <- ks.test(random_numbers, "punif")
> print(ks_test)

        Asymptotic one-sample Kolmogorov-Smirnov test

data:  random_numbers
D = 0.016747, p-value = 0.9418
alternative hypothesis: two-sided

> 
```

```
> expected_counts <- expected_counts / sum(expected_counts)
> chi_square_test <- chisq.test(table(cut(random_numbers, breaks = 20)), p = expec
ted_counts)
> print(chi_square_test)

        Chi-squared test for given probabilities

data:  table(cut(random_numbers, breaks = 20))
X-squared = 16.52, df = 19, p-value = 0.6224

> acf(random_numbers)
> 
```




```
(-0.000951,0.05]    (0.05,0.0999]    (0.0999,0.15]       (0.15,0.2]
              59               47               59               40
      (0.2,0.25]      (0.25,0.3]       (0.3,0.35]       (0.35,0.4]
              50               48               58               45
     (0.4,0.449]   (0.449,0.499]    (0.499,0.549]    (0.549,0.599]
              47               55               52               39
   (0.599,0.649]   (0.649,0.699]    (0.699,0.749]    (0.749,0.799]
              58               53               44               38
   (0.799,0.849]   (0.849,0.899]    (0.899,0.949]        (0.949,1]
              54               47               53               54
>
```

RESULT:

Thus, testing random numbers and random variates for their independence is done using R programming.

Ex.No:14	SOLVING SINGLE SERVER QUEUING MODEL USING SIMULATION SOFTWARE PACKAGE

AIM :

To solve single server queuing model using simulation software package.

PROCEDURE:

1. Problem Definition: Define the single server queuing model, including arrival rates, service rates, and queue characteristics.

2. Simulation Setup: Input the queuing model parameters into simulation software like Simul8 or Arena.

3. Simulation Execution: Run the simulation to model the queuing system's behavior over time.

4. Result Analysis: Analyze the simulation output to understand queue lengths, wait times, and system performance metrics.

OUTPUT:

QUEUEING MODELS

QUEUEING OUTPUT ANALYSIS

Select Output Option: Scenario1

Next Iteration | All Iterations | Write to Printer

Title: single server

Scenario 1:(M/M/1):(GD/infinity/infinity)

Lambda =	24.00000	Mu =	30.00000
L'da eff =	24.00000	Rho/c =	0.80000
Ls =	4.00000	Lq =	3.20000
Ws =	0.16667	Wq =	0.13333

n	Probability, pn	Cumulative, Pn	n	Probability, pn	Cumulative, Pn
0	0.20000	0.20000	23	0.00118	0.99528
1	0.16000	0.36000	24	0.00094	0.99622
2	0.12800	0.48800	25	0.00076	0.99698
3	0.10240	0.59040	26	0.00060	0.99758
4	0.08192	0.67232	27	0.00048	0.99807
5	0.06554	0.73786	28	0.00039	0.99845
6	0.05243	0.79028	29	0.00031	0.99876
7	0.04194	0.83223	30	0.00025	0.99901
8	0.03355	0.86578	31	0.00020	0.99921
9	0.02684	0.89263	32	0.00016	0.99937
10	0.02147	0.91410	33	0.00013	0.99949
11	0.01718	0.93128	34	0.00010	0.99959
12	0.01374	0.94502	35	0.00008	0.99968
13	0.01100	0.95602	36	0.00006	0.99974
14	0.00880	0.96482	37	0.00005	0.99979
15	0.00704	0.97185	38	0.00004	0.99983
16	0.00563	0.97748	39	0.00003	0.99987
17	0.00450	0.98199	40	0.00003	0.99989

View/Modify Input Data | MAIN Menu | Exit TORA

QUEUEING MODELS

QUEUEING OUTPUT ANALYSIS

Select Output Option: Scenario2

Next Iteration | All Iterations | Write to Printer

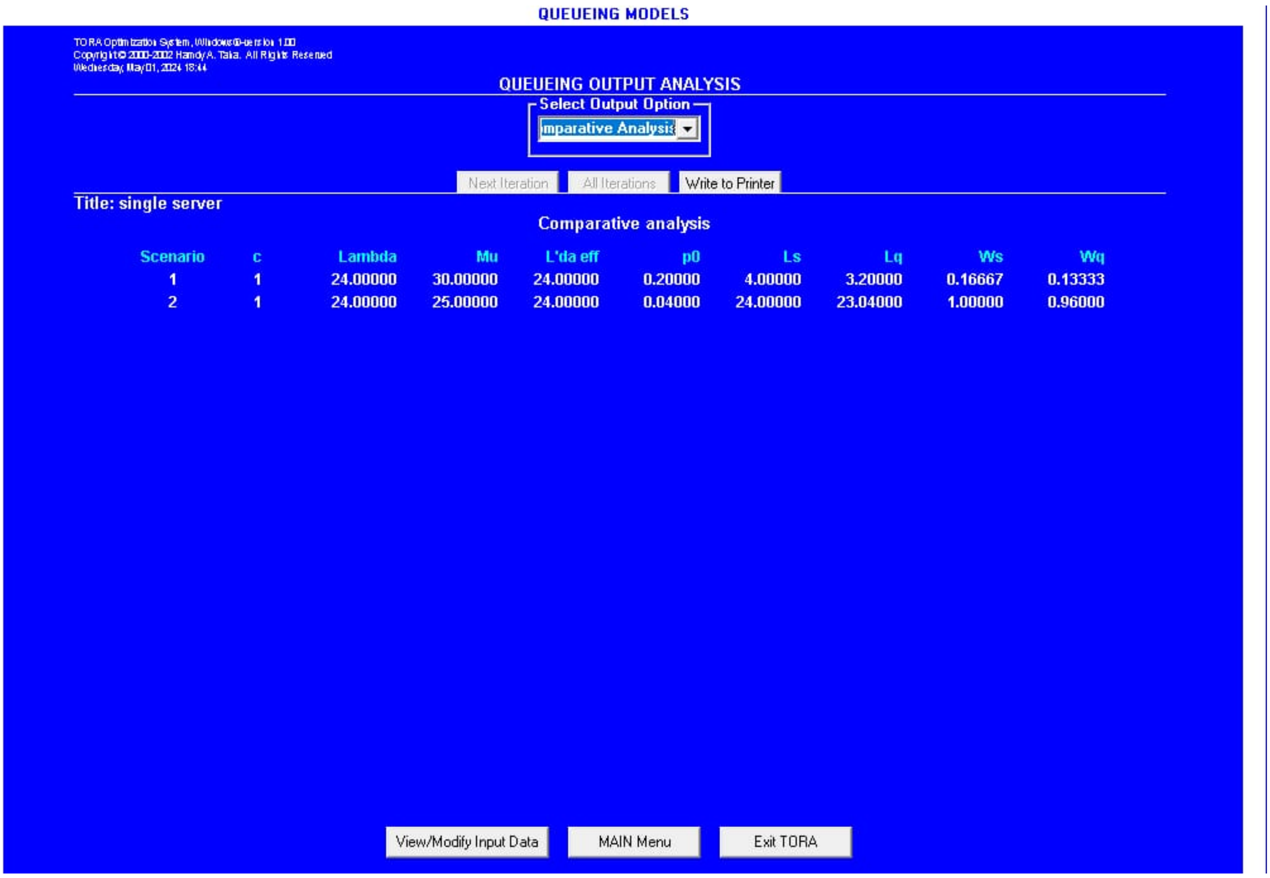
Title: single server

Scenario 2:(M/M/1):(GD/infinity/infinity)

Lambda =	24.00000	Mu =	25.00000
L'da eff =	24.00000	Rho/c =	0.96000
Ls =	24.00000	Lq =	23.04000
Ws =	1.00000	Wq =	0.96000

n	Probability, pn	Cumulative, Pn	n	Probability, pn	Cumulative, Pn
0	0.04000	0.04000	102	0.00062	0.98507
1	0.03840	0.07840	103	0.00060	0.98567
2	0.03686	0.11526	104	0.00057	0.98624
3	0.03539	0.15065	105	0.00055	0.98679
4	0.03397	0.18463	106	0.00053	0.98732
5	0.03261	0.21724	107	0.00051	0.98783
6	0.03131	0.24855	108	0.00049	0.98832
7	0.03006	0.27861	109	0.00047	0.98878
8	0.02886	0.30747	110	0.00045	0.98923
9	0.02770	0.33517	111	0.00043	0.98966
10	0.02659	0.36176	112	0.00041	0.99008
11	0.02553	0.38729	113	0.00040	0.99047
12	0.02451	0.41180	114	0.00038	0.99085
13	0.02353	0.43533	115	0.00037	0.99122
14	0.02259	0.45791	116	0.00035	0.99157
15	0.02168	0.47960	117	0.00034	0.99191
16	0.02082	0.50041	118	0.00032	0.99223
17	0.01998	0.52040	119	0.00031	0.99254

View/Modify Input Data | MAIN Menu | Exit TORA



Scenario	c	Lambda	Mu	L'da eff	p0	Ls	Lq	Ws	Wq
1	1	24.00000	30.00000	24.00000	0.20000	4.00000	3.20000	0.16667	0.13333
2	1	24.00000	25.00000	24.00000	0.04000	24.00000	23.04000	1.00000	0.96000

RESULT:

Thus , single server queuing model is solved using simulation software package.

Ex.No:15	SOLVING MULTI SERVER QUEUING MODEL USING SIMULATION SOFTWARE PACKAGE

AIM :

To solve single server queuing model using simulation software package.

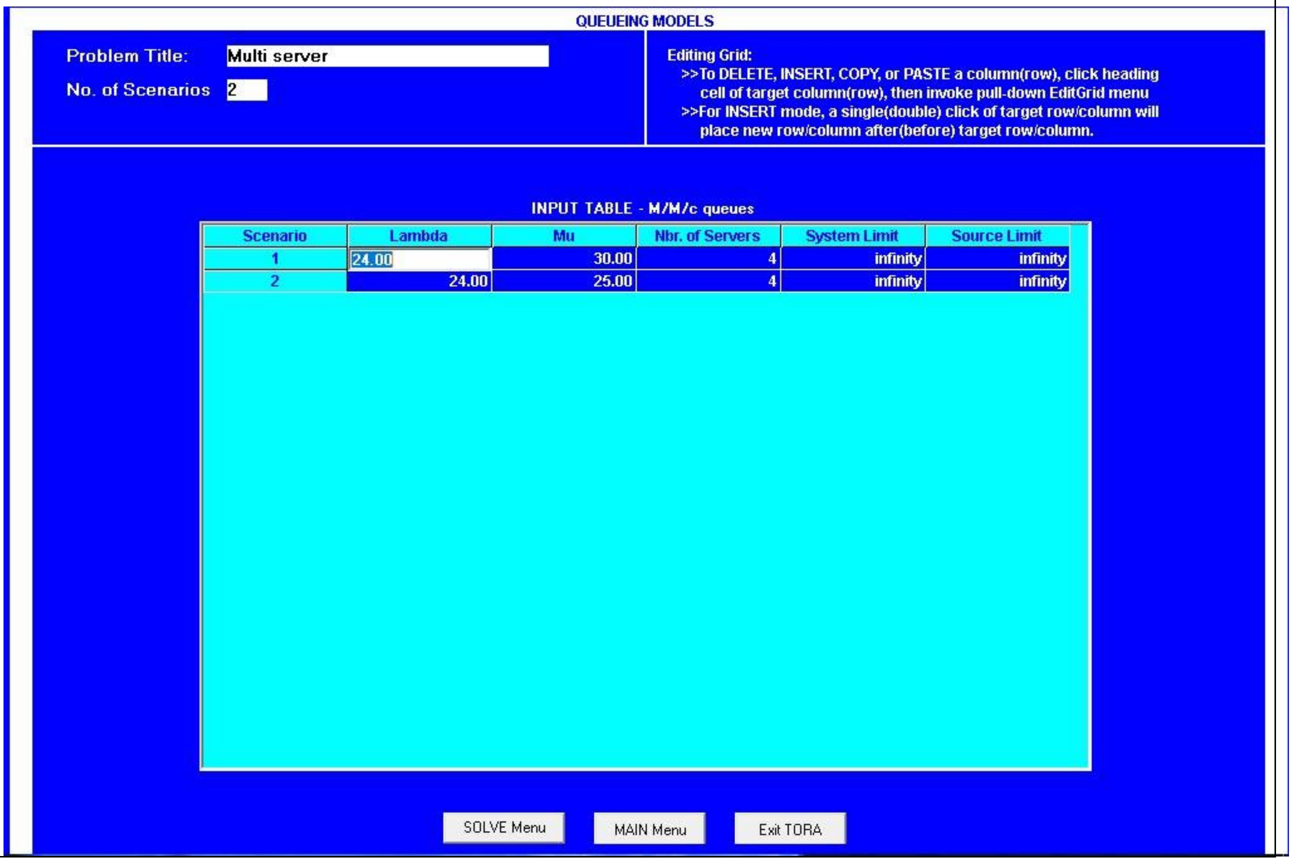
PROCEDURE:

1.Problem Definition: Define the multi-server queuing model, including parameters such as arrival rates, service rates for each server, and the number of servers.

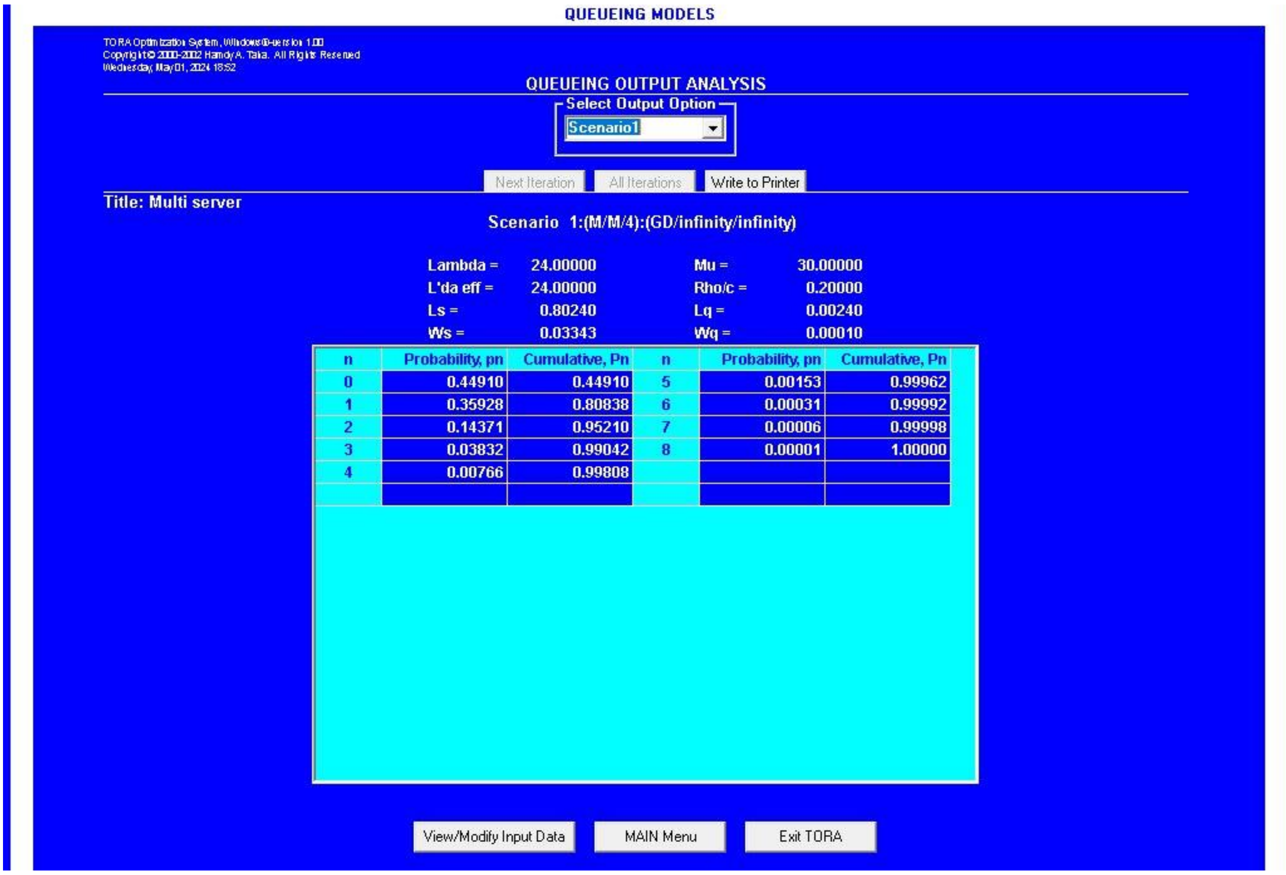
2.Simulation Setup: Input the queuing model parameters into simulation software like Simul8 or Arena, specifying the number of servers and their respective service rates.

3.Simulation Execution: Run the simulation to model the behavior of the multi-server queuing system over time. The simulation will track queue lengths, wait times, and system performance metrics as customers move through the system.

4.Result Analysis: Analyze the simulation output to evaluate the efficiency of the multi-server queuing system. This includes examining queue lengths, average wait times, server utilization, and overall system performance. Adjustments to the number of servers or their service rates can be made based on the simulation results to optimize system performance.



OUTPUT:



QUEUEING MODELS

TORA Optimization System, Windows®-version 1.00
Copyright© 2000-2002 Hamdy A. Taha. All Rights Reserved
Wednesday, May 01, 2024 18:52

QUEUEING OUTPUT ANALYSIS

Select Output Option: Scenario1

Next Iteration | All Iterations | Write to Printer

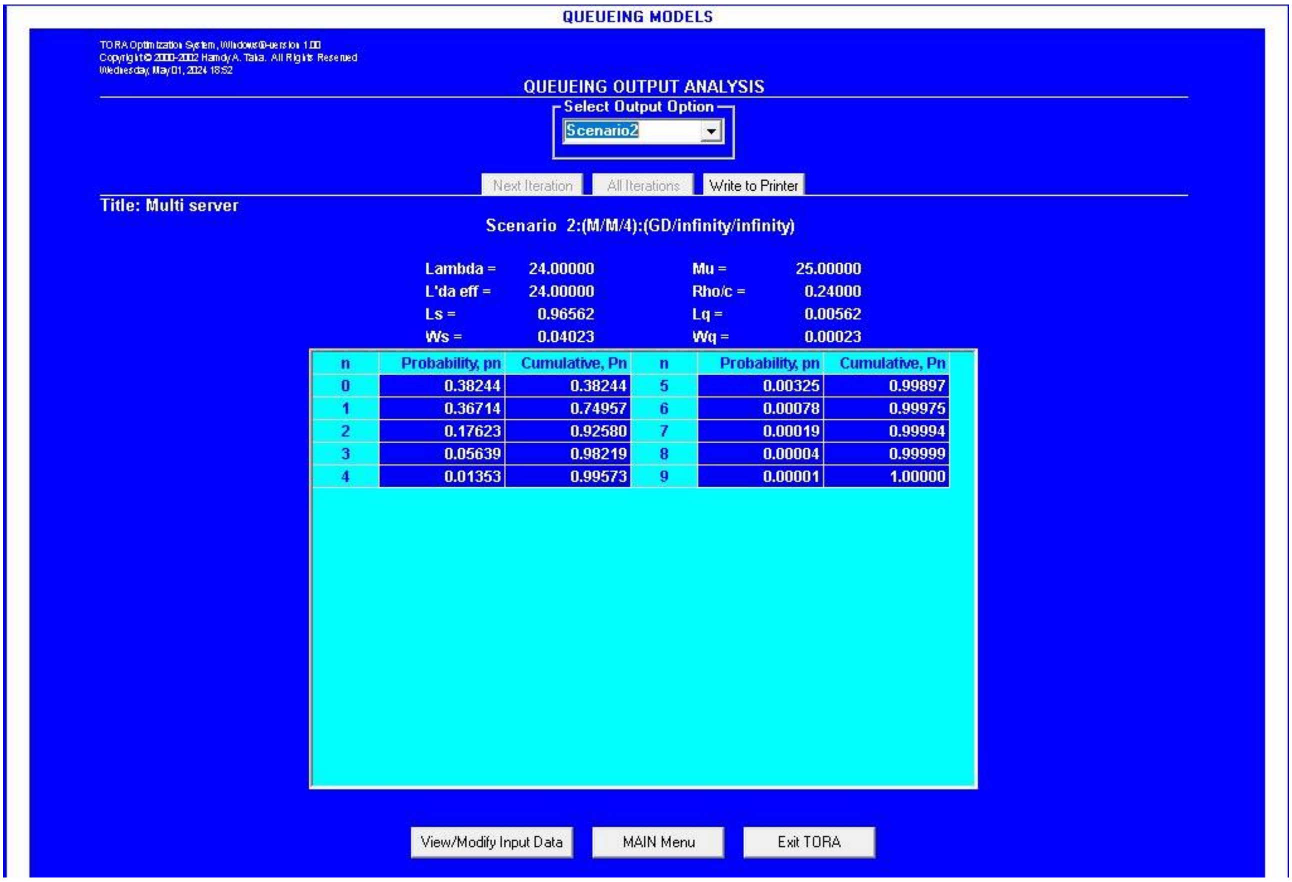
Title: Multi server

Scenario 1:(M/M/4):(GD/infinity/infinity)

Lambda =	24.00000	Mu =	30.00000
L'da eff =	24.00000	Rho/c =	0.20000
Ls =	0.80240	Lq =	0.00240
Ws =	0.03343	Wq =	0.00010

n	Probability, pn	Cumulative, Pn	n	Probability, pn	Cumulative, Pn
0	0.44910	0.44910	5	0.00153	0.99962
1	0.35928	0.80838	6	0.00031	0.99992
2	0.14371	0.95210	7	0.00006	0.99998
3	0.03832	0.99042	8	0.00001	1.00000
4	0.00766	0.99808			

View/Modify Input Data | MAIN Menu | Exit TORA



QUEUEING MODELS

TORA Optimization System, Windows®-version 1.00
Copyright© 2000-2002 Hamdy A. Taha. All Rights Reserved
Wednesday, May 01, 2024 18:52

QUEUEING OUTPUT ANALYSIS

Select Output Option: Scenario2

Next Iteration | All Iterations | Write to Printer

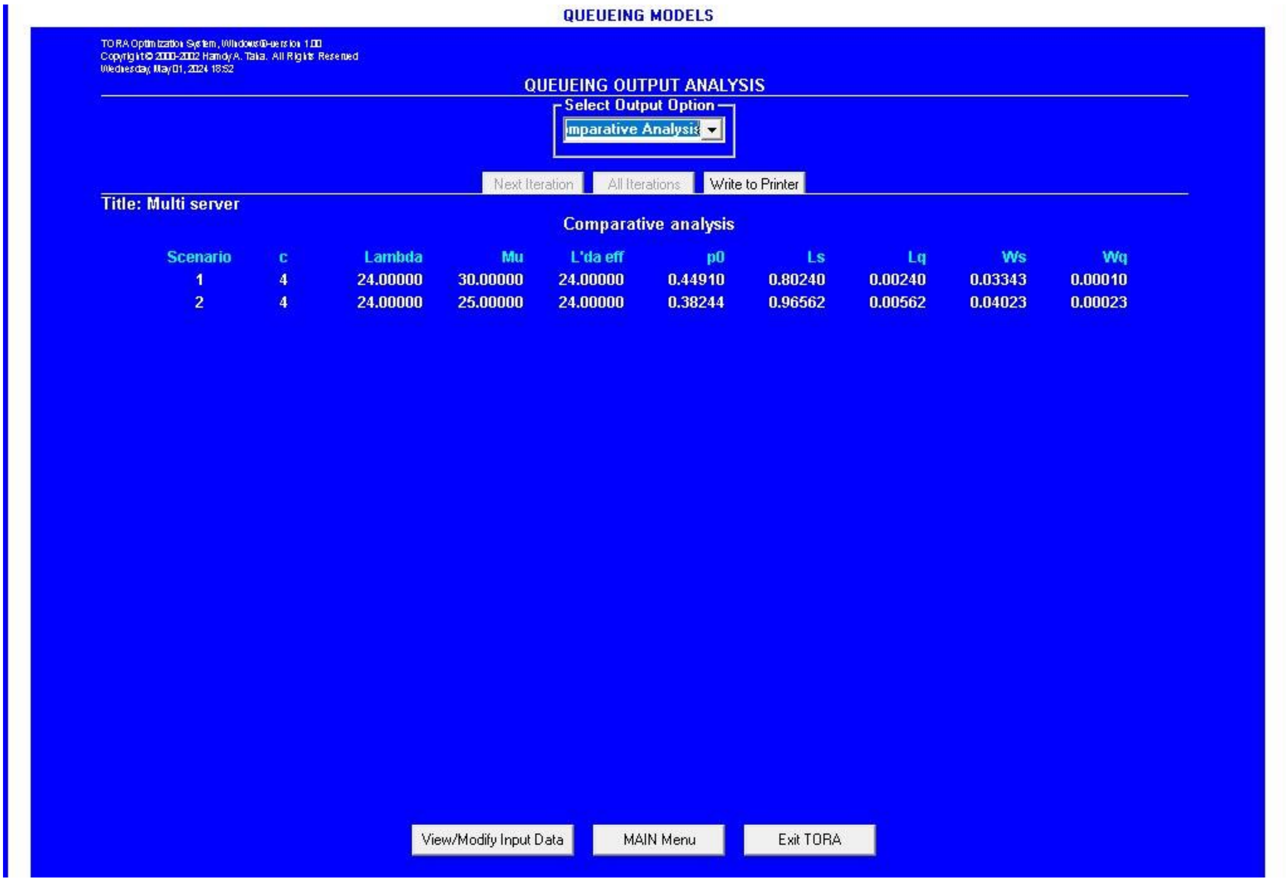
Title: Multi server

Scenario 2:(M/M/4):(GD/infinity/infinity)

Lambda =	24.00000	Mu =	25.00000
L'da eff =	24.00000	Rho/c =	0.24000
Ls =	0.96562	Lq =	0.00562
Ws =	0.04023	Wq =	0.00023

n	Probability, pn	Cumulative, Pn	n	Probability, pn	Cumulative, Pn
0	0.38244	0.38244	5	0.00325	0.99897
1	0.36714	0.74957	6	0.00078	0.99975
2	0.17623	0.92580	7	0.00019	0.99994
3	0.05639	0.98219	8	0.00004	0.99999
4	0.01353	0.99573	9	0.00001	1.00000

View/Modify Input Data | MAIN Menu | Exit TORA



RESULT:

Thus , multi server queuing model is solved using simulation software package.