Table of Contents

1. **Testing of Random Numbers.**

A) **Testing of Uniformity.**

i. **Implementation of Kolmogorov-Smirnov Test (K-S Test).**

Code:

```
#include <stdio.h>
#include<conio.h>

double dplus(double num[], double d1[], int length, double n) {

   for (int i = 0; i < length; i++) {
      d1[i] = ((i + 1) / n) - num[i];
   }
   double d1max = d1[0];
   for (int i = 0; i < length; i++) {
      if (d1max <= d1[i]) {
         d1max = d1[i];
      }
   }
   printf("D+ = %f\n", d1max);
   return d1max;
}
double dminus(double num[], double d2[], int length, double n) {
   for (int i = 0; i < length; i++) {
      d2[i] = (num[i] - (i) / n);
   }
   double d2max = d2[0];
   for (int i = 0; i < length; i++) {
      if (d2max <= d2[i]) {
         d2max = d2[i];
      }
   }
printf("D- = %f\n", d2max);
   return d2max;
}
void kst(double num[], int length) {
   for (int i = 0; i < length; i++) {
      for (int j = i + 1; j < length; j++) {
         if (num[i] > num[j]) {
            double temp;
            temp = num[i];
            num[i] = num[j];
            num[j] = temp;
         }
```

```
        }
    }
    double d1[length];
    double d2[length];
    double n = (double) length;
    double d1max = dplus(num, d1, length, n);
    double d2max = dminus(num, d2, length, n);
    double dplus = d1max;
    double dminus = d2max;
    double d;
    if (dplus > dminus) {
        d = dplus;
        printf("D = %f\n", d);
    } else {
        d = dminus;
        printf("D = %f\n", d);
    }
}
int main() {
    printf("Kolmogorov Test\n");
    int n;
    double dvalue1;
    printf("Enter number of elements to compute for tets: \t");
    scanf("%d", &n);
    double num[n];
    double dp, dn;
    int i;
    for (i = 0; i < n; i++) {
        scanf("%lf", &num[i]);
    }
    kst(num, i);
}
```

Output:

```
E:\Suyash\BSc.CSIT\Sem5\SM\Lab\ks.exe
Kolmogorov Test
Enter number of elements to compute for tets:   5
0.44
0.81
0.14
0.05
0.93
D+ = 0.260000
D- = 0.210000
D = 0.260000

--------------------------------
Process exited after 20.68 seconds with return value 0
Press any key to continue . . .
```

ii. Implementation of Chi Square Test.

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define NUM_BINS 10 // number of bins for chi-square test

int main()
{
   int num_random, i;
   double rand_num, expected_frequency, chi_square, chi_critical;
   int observed_frequency[NUM_BINS] = {0};

   // ask user for total number of random numbers
   printf("Enter total number of random numbers to generate: ");
   scanf("%d", &num_random);

   // generate random numbers and count their frequencies in bins
   printf("\nRandom numbers generated:\n");
   for (i = 0; i < num_random; i++) {
      rand_num = (double) rand() / RAND_MAX;
      observed_frequency[(int)(rand_num * NUM_BINS)]++;
      printf("%f ", rand_num);
   }

   // calculate expected frequency and chi-square value
   expected_frequency = (double) num_random / NUM_BINS;
   chi_square = 0.0;
   for (i = 0; i < NUM_BINS; i++) {
      chi_square += pow(observed_frequency[i] - expected_frequency, 2) /
expected_frequency;
   }

   // calculate critical value of chi-square at 5% significance level
   chi_critical = 16.92; // from chi-square table with 9 degrees of freedom

   // print results
   printf("\n\nChi-square value: %f\n", chi_square);
   if (chi_square > chi_critical) {
      printf("The random numbers do not pass the chi-square test at 5%% significance
level.\n");
   } else {
```

```
        printf("The random numbers pass the chi-square test at 5%% significance level.\n");
    }

    return 0;
}
```

Output:

```
E:\Suyash\BSc.CSIT\Sem5\SM\Lab\chisq.exe
Enter total number of random numbers to generate: 5

Random numbers generated:
0.001251 0.563585 0.193304 0.808741 0.585009

Chi-square value: 9.000000
The random numbers pass the chi-square test at 5% significance level.

--------------------------------
Process exited after 1.191 seconds with return value 0
Press any key to continue . . .
```

B) Testing of Independence.

i. Implementation of Poker Test.

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include<time.h>

#define N 100   // Random numbers to be generated

int main() {
    // Initialize the frequency table
    int freq[10] = {0};
    srand(time(0));
    printf("The generated 100 random numbers are:\n");

    // Generate random 3-digit numbers and count the frequency of each digit
    for (int i = 0; i < N; i++) {
        int num = rand() % 900 + 100;   // Generate a random 3-digit number
        printf("%d\t",num);
        int d1 = num / 100;   // Extract the first digit
```

```
        int d2 = (num / 10) % 10;   // Extract the second digit
        int d3 = num % 10;   // Extract the third digit
        freq[d1]++;
        freq[d2]++;
        freq[d3]++;
    }
    printf("\n");

    // Compute the chi-square statistic
    double chi_square = 0;
    for (int i = 0; i < 10; i++) {
        double expected = N/10.0;   // Expected frequency of each digit
        double observed = freq[i];
        chi_square += (observed - expected)*(observed - expected) / expected;
    }

    // Print the results
    printf("Chi-square statistic: %f\n", chi_square);
    if (chi_square < 16.92) {
        printf("The test passed (i.e., the numbers appear to be independent).\n");
    } else {
        printf("The test failed (i.e., the numbers do not appear to be independent).\n");
    }

    return 0;
}
```

Output:

```
E:\Suyash\BSc.CSIT\Sem5\SM\Lab\pokertest.exe
The generated 100 random numbers are:
273     332     660     970     806     537     216     215     466     590     223     339     933     460     617
        822     152     118     413     724     230     524     548     903     511     802     719     108     175
        263     303     719     343     448     122     272     519     574     957     345     692     339     384
        250     628     711     381     656     782     861     477     833     648     843     291     799     565
        408     374     783     936     776     185     375     514     949     525     443     948     805     967
        569     661     322     251     500     801     521     533     854     318     142     669     448     377
        132     180     949     768     104     848     921     615     218     128     281     825     756     293
        740
Chi-square statistic: 429.000000
The test failed (i.e., the numbers do not appear to be independent).

--------------------------------
Process exited after 0.03314 seconds with return value 0
Press any key to continue . . .
```

2. **Implement Linear Congruential Method for random number generation.**

Code:

```
#include<stdio.h>
#define MAX 50
int main()
{
	int i, n, a, seed, c, m, r[MAX];
	printf("How many random numbers you want to generate: ");
	scanf("%d",&n);
	printf("Enter value of a: ");
	scanf("%d",&a);
	printf("Enter value of c: ");
	scanf("%d",&c);
	printf("Enter value of m: ");
	scanf("%d",&m);
	printf("Enter seed value: ");
	scanf("%d",&r[0]);
	for(i=1;i<=n;i++)
	{
		r[i]=(a*r[i-1]+c)%m;
	}
	printf("The random numbers are:\n");
	for(i=1;i<=n;i++)
	{
		printf("%f\n",(float)r[i]/m);
	}

}
```

Output:

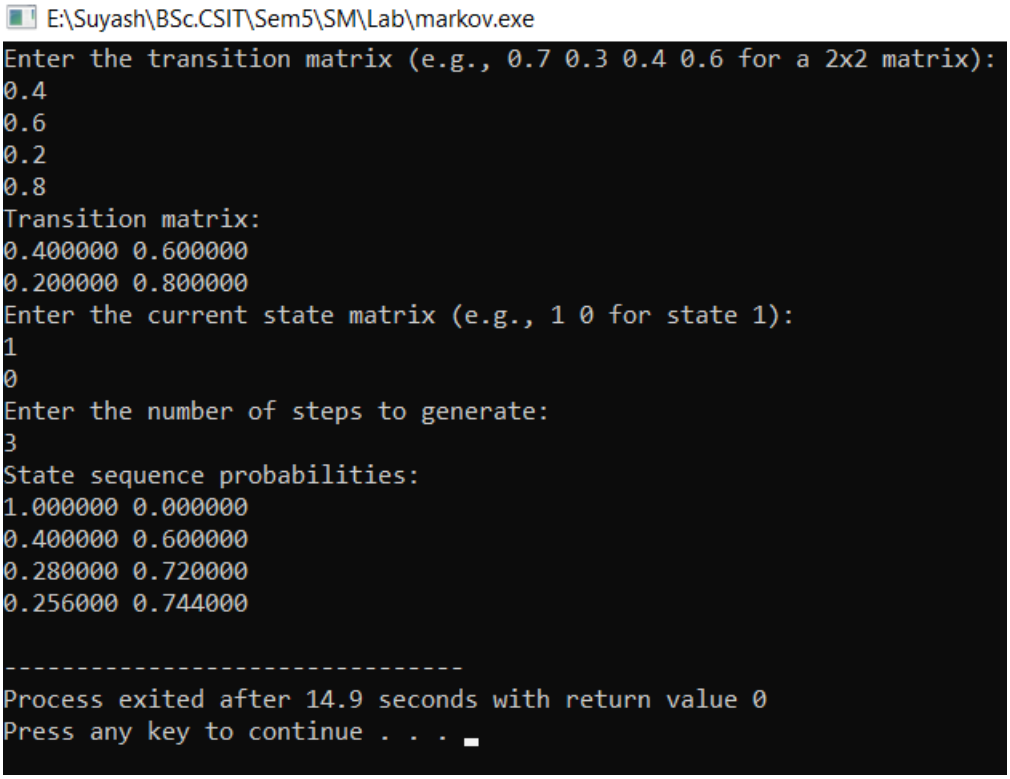
```
E:\Suyash\BSc.CSIT\Sem5\SM\Lab\random.exe
How many random numbers you want to generate: 5
Enter value of a: 27
Enter value of c: 4
Enter value of m: 100
Enter seed value: 23
The random numbers are:
0.250000
0.790000
0.370000
0.030000
0.850000

--------------------------------
Process exited after 12.86 seconds with return value 0
Press any key to continue . . .
```

3. Implement application of Markov Chain.

Code:

```
#include <stdio.h>
#include <stdlib.h>

int main() {
   // Define the transition matrix
   double P[2][2];
   printf("Enter the transition matrix (e.g., 0.7 0.3 0.4 0.6 for a 2x2 matrix):\n");
   scanf("%lf %lf %lf %lf", &P[0][0], &P[0][1], &P[1][0], &P[1][1]);

   // Print the transition matrix
   printf("Transition matrix:\n");
   printf("%f %f\n", P[0][0], P[0][1]);
   printf("%f %f\n", P[1][0], P[1][1]);

   // Define the current state matrix
   float x, y;
   printf("Enter the current state matrix (e.g., 1 0 for state 1):\n");
   scanf("%f %f", &x, &y);

   // Generate the Markov chain
   int n;
   printf("Enter the number of steps to generate:\n");
   scanf("%d", &n);

   printf("State sequence probabilities:\n");
   printf("%f %f\n", x, y);
   for (int i = 1; i <= n; i++) {
      // Generate the next state
      float new_x = P[0][0]*x + P[1][0]*y;
      float new_y = P[0][1]*x + P[1][1]*y;
      printf("%f %f\n", new_x, new_y);
      x = new_x;
      y = new_y;
   }

   return 0;
}
```

Output:



4. **Implement Monte-Carlo Method.**

Code:

```
/*Program to implement monte carlo method to determine the
probability of getting 3, 6 or 9 heads in 10 flips of a coin*/
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main()
{
   int i, j, n, h, t;
   double x;

   srand(time(NULL)); // seed random number generator

   // ask user for number of iterations
   printf("Enter number of iterations: ");
   scanf("%d", &n);

   // perform simulation
   t = h = 0;
```

```
    for (i = 0; i < n; i++) {
        h = 0;
        for (j = 0; j < 10; j++) {
            x = (double) rand() / RAND_MAX;
            if (x < 0.5) {
                printf("H ");
                h++;
            } else {
                printf("T ");
            }
        }
        printf("\n");
        if (h == 3 || h == 6 || h == 9) {
            t++;
        }
    }

    // calculate probability and display result
    double p = (double) t / n;
    printf("\nProbability of getting 3, 6, or 9 heads in 10 coin flips: %f\n", p);

    return 0;
}
```

Output:

```
E:\Suyash\BSc.CSIT\Sem5\SM\Lab\monte-carlo.exe
T T H T H T H T H H
T T H T H T H T H H
H H H T H T T T T T
T T T T T T H H H T
H H H T H T T H H H
H T H H T T T H H H
T H H T H T H T H H
T T H T T H T T T T
H H H H H T T H T H
T T H H T H T H T T
H T H T H H T T H H
T T T H T T T H T H
T H H T H T T T H H
T H T H H T T T H T
T H T T H H H T H T
T H T T T T H T T H
T T H T H T H T T T
H T T H H T H T H T
H H T H H H T T H H
H H H T H H H T T T
H H H T T H T T H T
H H H T T T H T T T
T T T H H T H H T T
T T H T H H H T T T

Probability of getting 3, 6, or 9 heads in 10 coin flips: 0.320000

--------------------------------
Process exited after 2.494 seconds with return value 0
Press any key to continue . . .
```

5. **Simulation of single server queue system using GPSS.**

   i. **Barber Shop simulation to simulate one day of operation of a barber.**

   Problem: Customers arrive at barber shop at the rate of 18 ± 6. Mechanic serves each customer at the rate of 16 ± 4 minutes. Simulate system for queue of customer and measure of waiting time for 25 customers.

   Model:

```
GENERATE 18,6          ;Customer arrive every 18+/-6 mn
QUEUE Chairs           ;Enter the line
SEIZE Joe              ;Capture the barber
DEPART Chairs          ;Leave the line
ADVANCE 16,4           ;Get a hair cut in 16+/-4 mn
RELEASE Joe            ;Free the barber
TERMINATE 1            ;Leave the shop
```

   Output:

```
                GPSS World Simulation Report - barbershop.3.1


                    Monday, March 13, 2023 12:36:09

          START TIME           END TIME  BLOCKS  FACILITIES  STORAGES
               0.000            466.570     7         1          0


             NAME                      VALUE
            CHAIRS                  10000.000
            JOE                     10001.000


 LABEL              LOC  BLOCK TYPE     ENTRY COUNT CURRENT COUNT RETRY
                    1    GENERATE            25             0       0
                    2    QUEUE               25             0       0
                    3    SEIZE               25             0       0
                    4    DEPART              25             0       0
                    5    ADVANCE             25             0       0
                    6    RELEASE             25             0       0
                    7    TERMINATE           25             0       0

 FACILITY         ENTRIES  UTIL.   AVE. TIME AVAIL. OWNER PEND INTER RETRY DELAY
  JOE                25    0.860      16.057  1        0    0    0     0       0

 QUEUE              MAX CONT. ENTRY ENTRY(0) AVE.CONT. AVE.TIME   AVE.(-0) RETRY
  CHAIRS              1    0     25     14     0.080      1.499      3.407   0

 FEC XN    PRI         BDT      ASSEM  CURRENT  NEXT  PARAMETER    VALUE
     26    0         467.353     26      0       1
```

ii. Barber Shop simulation (simple).

Problem: Customers arrive at barber shop at the rate of 300 ± 100. Mechanic serves each customer at the rate of 400 ± 200 minutes. Simulate system for queue of customer and measure of waiting time for 1000 customers.

Model:

```
GENERATE 300,100          ;Create next customer
QUEUE Barber              ;Begin queue time
SEIZE Barber              ;Own or wait for barber
DEPART Barber             ;End queue time
ADVANCE 400,200           ;Haircut takes a few minutes
RELEASE Barber            ;Haircut done. Give up the barber.
TERMINATE 1               ;Customer leaves
```

Output:

```
                GPSS World Simulation Report - barber(simple).3.1


                    Monday, March 13, 2023 12:48:47

          START TIME           END TIME  BLOCKS  FACILITIES  STORAGES
               0.000         401931.114     7         1          0


             NAME                       VALUE
           BARBER                     10000.000


 LABEL              LOC  BLOCK TYPE     ENTRY COUNT CURRENT COUNT RETRY
                    1    GENERATE          1347             0       0
                    2    QUEUE             1347           346       0
                    3    SEIZE             1001             1       0
                    4    DEPART            1000             0       0
                    5    ADVANCE           1000             0       0
                    6    RELEASE           1000             0       0
                    7    TERMINATE         1000             0       0

FACILITY         ENTRIES  UTIL.   AVE. TIME AVAIL. OWNER PEND INTER RETRY DELAY
 BARBER            1001    0.999     401.161  1     1001    0    0     0    346

QUEUE              MAX CONT. ENTRY ENTRY(0) AVE.CONT. AVE.TIME   AVE.(-0) RETRY
 BARBER            347  347   1347      1   174.888  52184.845  52223.615   0

CEC XN   PRI          M1      ASSEM  CURRENT  NEXT  PARAMETER    VALUE
  1001    0      298909.662   1001      3      4

FEC XN   PRI          BDT     ASSEM  CURRENT  NEXT  PARAMETER    VALUE
  1348    0      402257.310   1348      0      1
```

iii. Mechanic Shop simulation.

Problem: Customers arrive at mechanic shop at the rate of 300 ± 200. Mechanic serves each customer at the rate of 200 ± 50 minutes. Simulate system for queue of customer and measure of waiting time for 1000 customers.

Model:

```
GENERATE 300 200          ;create next customer
QUEUE MECHANIC
SEIZE MECHANIC            ;Own mechanic(resource)
DEPART MECHANIC
ADVANCE 200 50            ;Mechanic Takes some Time
RELEASE MECHANIC          ;Release the resource
TERMINATE 1               ;end simulation
```

Output:

```
            GPSS World Simulation Report - mechanicshop.4.1


                 Monday, March 13, 2023 12:51:23

          START TIME           END TIME  BLOCKS  FACILITIES  STORAGES
               0.000         302200.785     7        1          0


             NAME                      VALUE
          MECHANIC                  10000.000


 LABEL              LOC  BLOCK TYPE     ENTRY COUNT CURRENT COUNT RETRY
                    1    GENERATE          1000             0       0
                    2    QUEUE             1000             0       0
                    3    SEIZE             1000             0       0
                    4    DEPART            1000             0       0
                    5    ADVANCE           1000             0       0
                    6    RELEASE           1000             0       0
                    7    TERMINATE         1000             0       0

FACILITY         ENTRIES  UTIL.   AVE. TIME AVAIL. OWNER PEND INTER RETRY DELAY
 MECHANIC          1000    0.660     199.321  1        0    0    0     0      0

QUEUE              MAX CONT. ENTRY ENTRY(0) AVE.CONT. AVE.TIME   AVE.(-0) RETRY
 MECHANIC            2    0   1000    727     0.061     18.521     67.841   0

FEC XN    PRI         BDT      ASSEM  CURRENT  NEXT  PARAMETER    VALUE
  1001     0      302264.348   1001      0      1
```

iv. Telephone System Simulation.

Problem: A simple telephone system has two external lines. Calls, which originate externally, arrive every 100±60 seconds. When the line is occupied, the caller redials after 5±1 minutes have elapsed. Call duration is 3±1 minutes. A tabulation of the distribution of the time each caller takes to make a successful call is required. How long will it take for 200 calls to be completed?

Model:

```
Sets        STORAGE     2
Transit     TABLE       M1,.5,1,20      ;Transit times
            GENERATE    1.667,1         ;Calls arrive
Again       GATE SNF    Sets,Occupied   ;Try for a line
            ENTER       Sets            ;Connect call
            ADVANCE     3,1             ;Speak for 3+/-1 min
            LEAVE       Sets            ;Free a line
            TABULATE    Transit         ;Tabulate transit time
            TERMINATE   1               ;Remove a transaction
Occupied    ADVANCE     5,1             ;Wait 5 minutes
            TRANSFER    ,Again          ;Try again
```

Output:

```
                GPSS World Simulation Report - telephone.1.1


                    Monday, March 13, 2023 12:56:09

 START TIME          END TIME  BLOCKS  FACILITIES  STORAGES
      0.000           359.156     9        0          1


    NAME                        VALUE
AGAIN                           2.000
OCCUPIED                        8.000
SETS                        10000.000
TRANSIT                     10001.000


 LABEL              LOC  BLOCK TYPE     ENTRY COUNT CURRENT COUNT RETRY
                    1    GENERATE           216             0       0
AGAIN               2    GATE               762             0       0
                    3    ENTER              201             0       0
                    4    ADVANCE            201             1       0
                    5    LEAVE              200             0       0
                    6    TABULATE           200             0       0
                    7    TERMINATE          200             0       0
OCCUPIED            8    ADVANCE            561            15       0
                    9    TRANSFER           546             0       0


STORAGE             CAP. REM. MIN. MAX.  ENTRIES AVL.  AVE.C. UTIL. RETRY DELAY
 SETS                 2    1    0     2      201   1    1.677  0.839    0     0
```

TABLE	MEAN	STD.DEV.	RANGE			RETRY	FREQUENCY	CUM.%
TRANSIT	14.268	17.274				0		
			1.500	-	2.500		20	10.00
			2.500	-	3.500		41	30.50
			3.500	-	4.500		24	42.50
			4.500	-	5.500		0	42.50
			5.500	-	6.500		2	43.50
			6.500	-	7.500		9	48.00
			7.500	-	8.500		14	55.00
			8.500	-	9.500		12	61.00
			9.500	-	10.500		1	61.50
			10.500	-	11.500		0	61.50
			11.500	-	12.500		2	62.50
			12.500	-	13.500		9	67.00
			13.500	-	14.500		3	68.50
			14.500	-	15.500		1	69.00
			15.500	-	16.500		2	70.00
			16.500	-	17.500		4	72.00
			17.500	-	18.500		7	75.50
			18.500	-	_		49	100.00

FEC XN	PRI	BDT	ASSEM	CURRENT	NEXT	PARAMETER	VALUE
179	0	359.251	179	8	9		
196	0	359.367	196	8	9		
217	0	359.676	217	0	1		
161	0	359.972	161	8	9		
215	0	360.201	215	8	9		
167	0	360.580	167	8	9		
195	0	360.656	195	4	5		
197	0	360.826	197	8	9		
187	0	362.292	187	8	9		
210	0	362.403	210	8	9		
216	0	362.513	216	8	9		
205	0	363.396	205	8	9		
203	0	363.487	203	8	9		
211	0	363.852	211	8	9		
208	0	364.028	208	8	9		
207	0	364.827	207	8	9		
214	0	364.883	214	8	9		

v. Turnstile of Football Stadium.

Problem: Spectators arrive at a turnstile of a football stadium every 7±7 seconds and queue for admittance. The time to pass through is evenly distributed at 5±3 seconds. A model is required to determine the time taken by 300 people to pass through the turnstile.

Model:

```
In_use      EQU         5           ;Mean time
Range       EQU         3           ;Half range
            GENERATE    7,7         ;People arrive
            QUEUE       Turn        ;Enter queue
            SEIZE       Turn        ;Acquire turnstile
            DEPART      Turn        ;Depart the queue
            ADVANCE     In_use,Range ;Use turnstile
            RELEASE     Turn        ;Leave turnstile
            TERMINATE   1           ;One spectator enters
```

Output:

```
                GPSS World Simulation Report - Turnstil.1.1


                     Monday, March 13, 2023 13:08:33

 START TIME           END TIME  BLOCKS  FACILITIES  STORAGES
      0.000           2134.023     7        1          0


    NAME                        VALUE
IN_USE                           5.000
RANGE                            3.000
TURN                         10002.000


LABEL              LOC  BLOCK TYPE     ENTRY COUNT CURRENT COUNT RETRY
                   1    GENERATE           300             0       0
                   2    QUEUE              300             0       0
                   3    SEIZE              300             0       0
                   4    DEPART             300             0       0
                   5    ADVANCE            300             0       0
                   6    RELEASE            300             0       0
                   7    TERMINATE          300             0       0

FACILITY         ENTRIES  UTIL.   AVE. TIME AVAIL. OWNER PEND INTER RETRY DELAY
 TURN               300    0.690       4.906  1        0    0     0     0      0

QUEUE              MAX CONT. ENTRY ENTRY(0) AVE.CONT. AVE.TIME   AVE.(-0) RETRY
 TURN                3    0    300    150      0.319      2.270      4.540   0

FEC XN    PRI         BDT      ASSEM  CURRENT  NEXT  PARAMETER    VALUE
   301     0        2135.381    301      0      1
```

vi. Manufacturing Shop.

Problem: A machine tool in a manufacturing shop is turning out parts at the rate of every 5 minutes. As they are finished, the parts are turned over to an inspector who takes 4±3 minutes to examine each one and rejects about 10% of the parts as faulty. Each part will be represented by an transaction and the base time unit for the system is chosen as 1 minute. Simulate for 100 parts to leave the system.

Model:

```
        GENERATE 5
        QUEUE Insq
        ENTER Ins,1
        DEPART Insq
        ADVANCE 4,3
        LEAVE Ins,1
        TRANSFER 0.1,Acc,Rej
Acc     TERMINATE 0
Rej     TERMINATE 0
        GENERATE 480
        TERMINATE 1Ins     STORAGE 3
```

Output:

```
                GPSS World Simulation Report - manufacturing.1.1


                     Monday, March 13, 2023 14:25:03

              START TIME            END TIME  BLOCKS  FACILITIES  STORAGES
                   0.000           48000.000    11        0          1


                  NAME                         VALUE
              ACC                               8.000
              INS                           10000.000
              INSQ                          10001.000
              REJ                               9.000


 LABEL              LOC  BLOCK TYPE      ENTRY COUNT CURRENT COUNT RETRY
                    1    GENERATE           9600              0       0
                    2    QUEUE              9600              0       0
                    3    ENTER              9600              0       0
                    4    DEPART             9600              0       0
                    5    ADVANCE            9600              2       0
                    6    LEAVE              9598              0       0
                    7    TRANSFER           9598              0       0
ACC                 8    TERMINATE          8691              0       0
REJ                 9    TERMINATE           907              0       0
                   10    GENERATE            100              0       0
                   11    TERMINATE           100              0       0


QUEUE               MAX CONT. ENTRY ENTRY(0) AVE.CONT. AVE.TIME   AVE.(-0) RETRY
 INSQ                 1    0   9600   9600      0.000      0.000      0.000   0


STORAGE             CAP. REM. MIN. MAX.  ENTRIES AVL.  AVE.C. UTIL. RETRY DELAY
 INS                  3    1   0     2     9600   1    0.796  0.265    0     0


FEC XN    PRI         BDT      ASSEM  CURRENT  NEXT  PARAMETER    VALUE
  9699     0       48000.099   9699      5       6
  9700     0       48003.516   9700      5       6
  9701     0       48005.000   9701      0       1
  9702     0       48480.000   9702      0      10
```