ASSIGNMENT PROBLEM IN C

```
/* SOLVE AN ASSIGNMENT PROBLEM BY THE HUNGARIAN METHOD */
/* ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ */

#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <float.h>
#include <graphics.h>

#define MAX 7
#define YES 1
#define NO  0
#define M   10000;

typedef int array[MAX][MAX];
typedef int list[MAX];
void validity    (void);
void screen      (void);
void clear       (void);
void outline     (void);
void draw        (void);
void accept      (void);
void display     (void);
void hungarian   (void);
void solution    (void);
void accept_el   (void);
void modify      (void);
void fill        (void);
void reduce_first (void);
void reduce_again (void);
void assign      (void);
void lines       (void);
void check       (void);
void copy        (void);
void check_one   (void);
void problem     (void);
void algorithm   (void);
void problem_max_min(void);
void input_jobs (void);
void input_machines(void);

char  ch,ch1;
/*
int   i, j, k, l, m, n, r, c, row3, col, count, asgn_count, cost, index;
```

```
int  valid, min, flag, opt_flag, taken_flag;
int  mchn, job, d_job, d_mchn, r_count, c_count, row3_zero, col_zero ;
array value,temp, asgn, delete, temp1;
list r_index, c_index;*/

int  i, j, k, l, m, n, r, c, row, col, count, asgn_count, cost, index;
int  valid, min, flag, opt_flag, taken_flag;
int  mchn, job, d_job, d_mchn, r_count, c_count, row_zero, col_zero ;
array value, temp, asgn, delete, temp1;
list r_index, c_index;

void main()
 {
 // problem();
// algorithm();
 ch= 'y';
 while ( (ch == 'y') || (ch == 'Y') )
  {
  screen();
  accept();
  check();
  copy();
  if ( flag )
   display();
  modify();
  if ( flag )
   display();
  asgn_count = 0;
  reduce_first();
  display();
  hungarian();
  solution();
  gotoxy(2,24);
  clreol();
  gotoxy(79,24);
  printf("%c",186);
  gotoxy(2,24);
  printf("Do you want to continue with another problem ? [y/n] ");
  ch = getche();
  }
clrscr();
return;
}

void screen()
 {
```

```
clrscr();
gotoxy(1,1);
printf("%c",201);
for ( i = 2 ; i < 80 ; i++ )
 {
 gotoxy(i, 1);
 printf("%c",205);
 gotoxy(i, 5);
 printf("%c",205);
 gotoxy(i, 25);
 printf("%c",205);
 }
gotoxy(79,1);
printf("%c",187);
gotoxy(10,3);
printf(" TO SOLVE AN ASSIGNMENT PROBLEM BY THE HUNGARIAN
METHOD\n");
gotoxy(10,4);
printf("
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^\n ");
gotoxy(1,5);
printf("%c",204);
gotoxy(79,5);
printf("%c",185);
for ( i = 2 ; i < 25 ; i++ )
 {
 gotoxy(1,i);
 printf("%c",186);
 gotoxy(79,i);
 printf("%c",186);
 }
gotoxy(1,25);
printf("%c",200);
gotoxy(79,25);
printf("%c",188);
}


void clear()
 {
 gotoxy(2,24);
 printf("Press any key to continue....");
 getch();
 }
```

```
void accept()                          /* Accept all criteria of*/
 {                                             /* the problem                  */
  problem_max_min();
  screen();
  input_jobs();
  input_machines();
 // input_machines();

 d_mchn = 0;                           /* In the case of an unbalanced */
 d_job  = 0;                           /* problem, if the no. of jobs  */
 if ( job != mchn )                    /* is greater than the no. of   */
  if ( mchn > job )                    /* machines else create dummy   */
   d_job = mchn - job;                 /* jobs                         */
  else
   d_mchn = job - mchn;
 for ( i = 0 ; i < MAX ; i++ )
  for ( j = 0 ; j < MAX ; j++ )
   value[i][j] = 0;
 screen();
 draw();
 gotoxy(25, 6); printf("MACHINES");
 gotoxy(19,10); printf("J");
 gotoxy(19,11); printf("O");
 gotoxy(19,12); printf("B");
 if ( job > 2 )
  {
  gotoxy(19,13);
  printf("S");
  }
 accept_el();
 }

void validity()                        /* To check if the <enter>, <tab> */
 {                                     /* <backspace> or the <esc> key   */
 int key;                              /* was pressed before entering any*/
 do                                    /* value was entered              */
  {
  key = bioskey(1);
  if ( (key == 283) || (key == 7181) || (key == 19712) || (key == 19200) )
   {
   key = bioskey(0);
   key = 0;
   }
  }
```

```
  while (key == 0);
 }

void outline()
 {
 col = 0;
 for (i = 0; i < 4 ; i++ )
  {
  gotoxy(20 + i,7);
  printf("%c",205);
  row = 9 + job*2;
  gotoxy(20 + i,row);
  printf("%c",205);
  gotoxy(20 + i,9);
  printf("%c",196);
  }
 for (i = 0 ; i < mchn ; i++ )
  for ( j = 0 ; j < 5 ; j++ )
   {
   col++;
   gotoxy(24 + col,7);
   printf("%c",205);
   row = 9 + job*2;
   gotoxy(24 + col,row);
   printf("%c",205);
   gotoxy(24 + col,9);
   printf("%c",196);
   }
 for (i = 0; i < 2 ; i++ )
  {
  gotoxy(20,7+i);
  printf("%c",186);
  col = 24 + mchn*5;
  gotoxy(col,7+i);
  printf("%c",186);
  gotoxy(24,7+i);
  printf("%c",179);
  }
 row = 0;
 for (i = 0 ; i < job ; i++ )
  for ( j = 0 ; j < 2 ; j++ )
   {
   row++;
   gotoxy(20,9+row);
   printf("%c",186);
   col = 24 + mchn*5;
```

```
        gotoxy(col,9+row);
        printf("%c",186);
        gotoxy(24,9+row);
        printf("%c",179);
        }
    gotoxy(20,7);    printf("%c",201);
    col = 24 + mchn*5;
    gotoxy(col,7);   printf("%c",187);
    row = 9 + job*2;
    gotoxy(col,row); printf("%c",188);
    gotoxy(20,row);  printf("%c",200);
    gotoxy(24,7);    printf("%c",209);
    gotoxy(24,row);  printf("%c",207);

    gotoxy(20,9);    printf("%c",199);
    gotoxy(col,9);   printf("%c",182);
    gotoxy(24,9);    printf("%c",197);
    gotoxy(10,15);   printf(" J O B S ");
    gotoxy(30,6);    printf(" M A C H I N E S ");
    }

void draw()                          /* Draw the table */
    {
    outline();
    col = 24;
    for ( i = 0 ; i < mchn-1 ; i++ )
     {
     row = 9;
     col = col + 5;
     gotoxy(col,row);

     printf("%c",194);
     for ( j = 0 ; j < job ; j++ )
      for ( k = 0 ; k < 2 ; k++ )
          {
          row++;
          gotoxy(col,row);
          printf("%c",179);
          }
     gotoxy(col,row);
     printf("%c",207);
     }
row = 9;
for ( i = 0 ; i < job-1 ; i++ )
    {
    col = 24;
```

```
    row = row + 2;
    gotoxy(col,row);
    printf("%c",195);
    for ( j = 0 ; j < mchn ; j++ )
     {
     for ( k = 0 ; k < 5 ; k++ )
         {
         col++;
         gotoxy(col,row);
         printf("%c",196);
         }
     gotoxy(col,row);
     printf("%c",197);
     }
    gotoxy(col,row);
    printf("%c",182);
    }
   col = 21;
   for ( i = 0 ; i < mchn ; i++ )
    {
    col = col + 5;
    gotoxy(col, 8);
    printf("%d",i+1);
    }
   row = 8;

   for ( i = 0 ; i < job ; i++ )
    {
    row = row + 2;
    gotoxy(22, row);
    printf("%d",i+1);
    }
  }

 void accept_cl()
  {
  gotoxy(2,24);
  printf("Enter all the pre-assigned costs in the corresponding cells.");
  row = 8;
  for ( i = 0 ; i < job ; i++ )
   {
   row = row + 2;
   col = 20;
   for ( j = 0 ; j < mchn ; j++ )
    {
    col = col + 5;
```

```
    gotoxy(col,row);
    validity();
    scanf("%d",&value[i][j]);
    }
  }
 }

void check()                        /* If the objective is to maximise*/
 {                                    /* negate all the values in the  */
 flag = 0;                            /* table and proceed             */
 if ( (ch1 == 'a') || (ch1 == 'A') )
  {
  flag = 1;
  for ( i = 0 ; i < job ; i++ )
   for ( j = 0 ; j < mchn ; j++ )
       {
       temp[i][j] = value[i][j];
       value[i][j] = -value[i][j];
       }
  }
 }

void copy()                                 /* Store the original table */
 {
 for ( i = 0 ; i < job ; i++ )
  for ( j = 0 ; j < mchn ; j++ )
   temp[i][j] = value[i][j];
 }

void modify()                           /* If the problem is unbalanced */
 {                                    /* add dummy row33s or dummy col3umns */
 flag = 0;
 if ( (d_mchn != 0) || (d_job!= 0) )
  flag = 1;
 mchn = mchn + d_mchn;
 job  = job  + d_job;
 }

void display()
 {
 screen();
 draw();
 fill();
 }

void fill()
```

```
{
row = 8;
for ( i = 0; i < job ; i++ )
 {
 row = row + 2;
 col = 20;
 for ( j = 0 ; j < mchn ; j++ )
  {
  col = col + 5;
  gotoxy(col,row);
  printf("%d",temp[i][j]);
  }
 }
if ( asgn_count != 0 )
 {
 for ( j = 0 ; j < r_count; j++ )
  {
  i = r_index[j];
  row = 10 + (2*i);
  gotoxy(19, row);
  printf("%c",16);
  col = 25 + (5 * mchn);
  gotoxy(col, row);
  printf("%c",17);
  }
 for ( j = 0 ; j< c_count ; j++ )
  {
  i = c_index[j];
  col = 25 + (5*i);
  gotoxy(col, 6);
  printf("%c",31);
  row = 10 + (2 * job);
  gotoxy(col, row);
  printf("%c",30);
  }
 }
gotoxy(2,24);
printf("Press any key to continue....");
getch();
}

void hungarian()
 {
 assign();
 while ( asgn_count != job )        /* Assign jobs to the  */
  {                                  /* machines. If not optimal */
```

```
  lines();                          /* cross out all zeroes */
  display();                        /* with the minimum no. of lines */
  reduce_again();                   /* Find the minimum of all the */
  assign();                         /* uncrossed elements & subtract*/
  }                                 /* it from all the uncrossed  */
}
                                    /* at the intersection of the lines*/


void reduce_first()                 /* Done in the first iteration */
 {                                  /* Find the minimum of each row3 */
 for ( i = 0 ; i < job ; i++ )      /* Subtract it from all the ele- */
  {                                 /* ments of that row3 . Similarly */
  min = M;                          /* for col3umn                    */
  for ( j = 0 ; j < mchn ; j++ )
   {
   if ( temp[i][j] < min)
        min = temp[i][j];
   }
  for ( j = 0 ; j < mchn ; j++ )
   temp[i][j] = temp[i][j] - min:
  }
 for ( i = 0 ; i < mchn ; i++ )
  {
  min = M;
  for ( j = 0 ; j < job ; j++ )
   {
   if ( temp[j][i] < min )
        min = temp[j][i];
   }
  for ( j = 0 ; j < job ; j++ )
   temp[j][i] = temp[j][i] - min;
  }
 }

void reduce_again()          /* Find the minimum of */
 {                                  /* all uncrossed elements */
 min = M;                           /* and subtract it from the */
 for ( i = 0 ; i < job ; i++ )      /* rest of the uncrossed elements */
  {
  flag = 0;
  index = i;
  for ( k = 0 ; k < r_count ; k++ )
   if ( index == r_index[k] )
        flag = 1;
  if ( !flag )
```

```
    {
    for ( j = 0 ; j < mchn ; j++ )
        {
        flag = 0;
        index = j;
        for ( k = 0 ; k < c_count ; k++ )
         if ( index == c_index[k] )
           flag = 1;
        if ( !flag )
         {
         if ( temp[i][j] < min )
           min = temp[i][j];
         }
        }
   }
  }

 for ( i = 0 ; i < job ; i++ )
  {
  flag = 0;
  index = i;
  for ( k = 0 ; k < r_count ; k++ )
   if ( index == r_index[k] )
        flag = 1;
  if ( !flag )
   {
   for ( j = 0 ; j < mchn ; j++ )
        {
        flag = 0;
        index = j;
        for ( k = 0 ; k < c_count ; k++ )
         if ( index == c_index[k] )
           flag = 1;
        if ( !flag)
         temp[i][j] = temp[i][j] - min;
        }
   }
  }
 for ( i = 0 ; i < r_count ; i++ )
  {
  l = r_index[i];
  for ( j = 0 ; j < c_count ; j++ )
   {
   m = c_index[j];
   temp[l][m] = temp[l][m] + min;
   }
```

```
  }
 }

void assign()
 {
 for ( r = 0 ; r < job ; r++ )
  for ( c = 0 ; c < mchn ; c++ )
   {
   asgn[r][c] = -1;
   temp1[r][c] = temp[r][c];
   }
 asgn_count = 0;
 for ( r = 0 ; r < job ; r++ )
  for ( c = 0 ; c < mchn ; c++ )
   {
   check_one();
   if ( temp1[r][c] == 0 )
       {
       asgn[r][c] = 0;
       asgn_count++;
       for ( l = 0 ; l < job ; l++ )
        if (temp[l][c] == 0)
          temp1[l][c] = M;
       for ( m = 0 ; m < mchn ; m++ )
        if (temp1[r][m] == 0)
          temp1[r][m] = M;
       }
   }
 }

void check_one()
 {
 for ( k = 0 ; k < job ; k++ )
  {
  for ( i = 0 ; i < job ; i++ )
   {
   count = 0;
   for ( j = 0 ; j < mchn ; j++ )
      if ( temp1[i][j] == 0 )
        {
        col = j ;
        count++;
        }
   if ( count == 1 )
       {
       asgn[i][col] = 0;
```

```
        asgn_count++;
        for ( l = 0 ; l < job ; l++ )
         if ( (temp[l][col] == 0) && (asgn[l][col] != 0) )
           temp1[l][col] = m;
        for ( l = 0 ; l < mchn ; l++ )
         if (temp1[i][l] == 0)
           temp1[i][l] = M;
        }
   }
  for ( i = 0 ; i < mchn ; i++ )
   {
   count = 0;
   for ( j = 0 ; j < job ; j++ )
       if ( temp1[j][i] == 0 )
         {
         row = j ;
         count++;
         }
   if ( count == 1 )
        {
        asgn[row][i] = 0;
        asgn_count++;
        for ( l = 0 ; l < mchn ; l++ )
         if (temp[row][l] == 0)
           temp1[row][l] = M;
        for ( m = 0 ; m < job ; m++ )
         if ( temp1[m][i] == 0)
           temp1[m][i] = 0;
        }
    }
   }
}

void lines()
 {
 for ( i = 0 ; i < job ; i++ )
  for ( j = 0 ; j < mchn ; j++ )
    temp1[i][j] = temp[i][j];
 r_count = 0;
 c_count = 0;

 for ( i = 0 ; i < job ; i++ )
  for ( j = 0 ; j < mchn ; j++ )
    {
    if ( asgn[i][j] == 0 )
        {
```

```
        row_zero = 0;
        col_zero = 0;
        for ( l = 0 ; l < job ; l++ )
         if ( temp1[l][j] == 0 )
           col_zero++;
        for ( l = 0 ; l < mchn ; l++ )
         if ( temp1[i][l] == 0 )
           row_zero++;
        if ( row_zero > col_zero )
         {
         r_index[r_count] = i;
         r_count++;
         for ( l = 0 ; l < mchn ; l++ )
          if ( temp1[i][l] == 0 )
            temp1[i][l] = M;
         }
        else
         {
         c_index[c_count] = j;
         c_count++;
         for ( l = 0 ; l < job ; l++ )
          if ( temp1[l][j] == 0 )
            temp1[l][j] = M;
         }
        j = mchn - 1;
        }
   }

for ( i = job ; i > 0 ; i-- )
 {
 for ( j = 0 ; j < job ; j++ )
  {
  count = 0;
  for ( k = 0 ; k < mchn ; k++ )
      if ( temp1[j][k] == 0 )
        count++;
  if ( count == i )
      {
      r_index[r_count] = j;
      r_count++;
      for ( l = 0 ; l < mchn ; l++ )
       if ( temp1[j][l] == 0 )
         temp1[j][l] = M;
      }
  }
for ( j = 0 ; j < mchn ; j++ )
```

```
    {
    count = 0;
    for ( k = 0 ; k < job ; k++ )
        if ( temp1[k][j] == 0 )
          count++;

    if ( count == i )
        {
        c_index[c_count] = j;
        c_count++;
        for ( l = 0 ; l < job ; l++ )
          if ( temp1[l][j] == 0 )
            temp1[l][j] = M;
        }
    }
  }
}

void solution()
 {
 lines();
 display();
 screen();
 gotoxy(3,6);
 printf(" Solution : ");
 gotoxy(3,8);
 printf(" The assignment can be done as follows. ");
 row = 9 ;
 col = 5;
 for ( i = 0 ; i < (job - d_job) ; i++ )
  for ( j = 0 ; j < (mchn - d_mchn) ; j++ )
   if ( asgn[i][j] == 0 )
        {
        row++;
        gotoxy(col,row);
        printf("job %d can be assigned to machine %d.",i+1,j+1);
        }
 cost = 0;
 for ( i = 0 ; i < job ; i++ )
  for ( j = 0 ; j < mchn ; j++ )
   if ( asgn[i][j] == 0 )
        cost = cost + value[i][j];
 row = row + 2;
 gotoxy(col,row);
 if ( (ch1 == 'i') || (ch1 == 'I') )
  printf(" The minimal cost of the entire operation is estimated as %d.",cost);
```

```
  else
   {
   cost = -cost;
   printf(" The maximal cost of the entire operation is estimated as %d.",cost);
   }
 }

void problem_max_min()
{
 gotoxy(3,6);
  printf(" Is the assignment problem a  ");
  gotoxy(5,8);
  printf(" A> Maximization problem  ");
  gotoxy(5,9);
  printf(" I> Minimization problem  ");
  gotoxy(3,11);
  printf(" (Press 'a' for a maximization problem ");
  gotoxy(3,12);
  printf("    and 'i' for a minimization one ) ");
  valid = NO;
  while ( valid != YES )
   {
   gotoxy(45, 12);
   ch1 = getche();
   if ( (ch1 == 'a') || (ch1 == 'i') || (ch1 == 'A') || (ch1 == 'I') )
    valid = YES;
   else
    {
    gotoxy(2,24);
    printf("  Wrong key !  Press the right one < 'a' or 'i' >");
    }
   }
}

void input_jobs()
{
  valid = NO;
  while ( valid != YES )
  {
  gotoxy(3,6);
  printf("  Number of jobs (rows) :  ");
  gotoxy(2,24);
  printf("  Number of jobs cannot be > 7 ! ");
  gotoxy(40,6);
  validity();
  scanf("%d",&job);
```

```
if ( ( job > 0 ) && ( job <= 7 ) )
 {
 for ( i = 2 ; i < 60 ; i++ )
 {
     gotoxy(i,24);
     printf(" ");
 }
 valid = YES;
 }

 else
 {
 row = 6;
 col = 40;
 for ( i = 0; i < 10 ; i++ )
     {
     gotoxy(col+i,row);
     printf(" ");
     }
 }
 }
}
void input_machines()
 {
 valid = NO;
 while ( valid != YES )
 {
 gotoxy(3,8);
 printf("  Number of machines (columns) :  ");
 gotoxy(2,24);
 printf("  Number of machines cannot be > 7 ! ");
 gotoxy(40,8);
 validity();

 scanf("%d",&mchn);
 if ( (mchn > 0) && (mchn <= 7) )
 {
 for ( i = 2 ; i < 60 ; i++ )
     {
     gotoxy(i,24);
     printf(" ");
     }
 valid = YES;
 }
 else
```

```
    {
    row = 8;
    col = 40;
    for ( i = 0; i < 10 ; i++ )
        {
        gotoxy(col+i,row);
        printf(" ");
        }
    }
  }
}
```

OUTPUT:

TO SOLVE AN ASSIGNMENT PROBLEM BY THE HUNGARIAN METHOD
^^

Is the assignment problem a

A> Maximization problem
I> Minimization problem

(Press 'a' for a maximization problem and 'i' for a minimization one) I (if) ↵

Number of jobs (rows) : 4 ↵

Number of machines (columns) : 4 ↵

MACHINES

JOBS	1	2	3	4
1	8	26	17	11
2	2	13	4	26
3	38	19	18	15
4	19	26	24	10

press any key to continue ↵

MACHINES

JOBS	MACHINES 1	2	3	4
1	0	14	9	3
2	9	20	0	22
3	23	0	3	0
4	9	12	14	0

Press any key to continue....

Solution

The assignment can be done as follows.

job 1 can be assigned to machine 1.
job 2 can be assigned to machine 3.
job 3 can be assigned to machine 2.
job 4 can be assigned to machine 4.

The minimal cost of the entire operation is estimated as 41.

Do you want to continue with another problem ? [y/n]

SPECIAL NOTE ABOUT THE ASSIGNMENT PROGRAM

This Program is free from all the errors. and can be easily execute in the computer.

We are using modular approach in this program.

This program contains maximum seven (7) rows and maximum (7) columns.

When you put the values in table according to problem, you should press the ENTER key for substituting the values row by row.

This program is applicable for both the minimization or maximization and balaned or unbalanced problems.

Travelling –salesman problem (routing) problem cannot be solve by this program. but later, in the next edition, we will provide travelling salesman problem program because this is also important of assignment problem.

Your suggestions or comments are very necessary for modifying this Program.

—AUTHOR

INVENTORY CONTROL C PROGRAMS

Inventory is nothing but the stock of goods, commodities or other resources that are stored for future use. Inventory is an important part of Organisation engaged in the production of goods & services.

Main Objective of Inventory Control is to find the order quantity which is most economical from the point of view operation.

Whole description of inventory control is given in out book.

I have used Costs in these programs with symbols are

(1) Ordering cost Co
(2) Carrying Cost Cc
(3) Setup cost Cs
(4) Shortage Cost Cs
(5) Unit cost Cu

I am giving you four inventory models program which are free from all the errors and easily execute in the computers.

Similarly on the basis of these programs you can solve any type of inventory control problems with the help of C language.

In inventory problems, you realized annual demand is given in most of the questions are given in per week or per month or per day. You should cahnge annual demand in per year.

If demand is given for week suppose it is x

Annual Demand D = x * 52 = 52 x per year (1 year = 52 weeks)

If demand is given for month suppose it is x

Annual Demand D = x * 12 = 12 x per year (1 year = 12 months)

If demand is given for day suppose it is x

Annual Demand D = x * 365 = 365 x per year (1 year = 365 days)

I have made 4 Inventory models C Programs on the basis of four examples. these are given below :

Example 1 (Program–1)

A Stockist has to supply 500 units of a product to his customers . he gets the product at Rs. 50 per unit from the manufacture. The cost of ordering and transportation from the manufacture is Rs. 75 per order. the cost of carrying inventory is 7.5%. per year of the cost of the product.

(1) what is the economic lot size EOQ ?
(2) How long would it take to produce economic lot size ?
(3) What is the total optimum cost per week ?

Read this problem and put all the values in program –1 and for entering the values you should press the ENTER key.

Example 2 (Program–2)

Dr Reddy's laboratories requires 1000 units of a particular drug additives per month; the average demand occurs at the rate of 30 units per day. The production process is capable of producing 50 units per day. Each item produced in the laboratory cost rupees 10. the set up cost per order is Rs. 100/-. The inventory carrying cost is 15% of the average inventory cost. Calculate

(1) the quantity to be produced in each production run
(2) no of production runs per year
(3) time interval between each production run
(4) total annual cost in cluding cost of drug

Read this problem and put all the values in program –2 and for entering the values you should press the ENTER key.

Example 3 (Program–3)

A Company is to be supplied at a constant rate of 200 units per day and production rate/day is 400 units. the supplies of any amount can be head at any required time but each ordering cost is Rs. 10/-. The cost of holding the commodity in the inventory is Rs. 2/- per unit/day while the delay in the supply of the items induces a penalty of Rs. 10/- unit/day. Find the optimum policy (Q,T) Where Q is the inventory level after reorder.
Read this problem and put all the values in program –3 and for entering the values you should press the ENTER key.

Example 4 (Program–4)

The annual demand for a product is 5400 units. Ordering Cost is Rs. 600/- per order.
Inventory carrying cost is 30% of the purchase cost per unit per year. The price breaks are given below

Quantity	Price/unit (Rs.)
0<= Q <= 2400	12
2400 <= Q <= 3000	10
Q >= 3000	08

In this program I have taken quantity Q1, Q2, Q3 according table which is shown below :

Q1 <= Q <=Q2
Q2 <= Q <=Q3
Q >= Q3

Similarly you can find more results with the help of this program when you substitute any value.

INVENTORY CONTROL

PROGRAM–1 PROGRAM WITH UNIFORM RATE OF DEMAND

PROGRAM 2– PRODUCTION MODEL

```
/* TO SOLVE THE PROBLEM OF INVENTORY LEVEL MODEL2
PRODUCTION MODEL*/
#include<stdio.h>
#include<math.h>

void main()
{
float d1,d2,d,h,P,D,Cc,Co,Cu,i,i1,Q,N,t,T,Tc,TAC,EPQ;
clrscr();
printf("enter the value of given demand per month d1 and daily
demand(consumption )rate d per day \n");
scanf("%f%f",&d1,&d2);
D=d1*12;
d=d2;
printf(" The annual demand D is=%0.2f per year\n",D);
printf("daily demand=%0.2f per day\n",d);
printf(" Enter the Production rate P per day  according to problem\n");
scanf("%f",&P);
printf("Enter the Set up cost Co , Unit cost Cu and rate of interest i1 (interest
percentage) according to given problem \n");
scanf("%f%f%f",&Co,&Cu,&i1);
i=(i1/100);
Cc=Cu*i;
h=1-(d/P);
EPQ=sqrt((2*D*Co)/(Cc*h));
N=D/EPQ;
t=365/N;
Tc=sqrt(2*D*Co*Cc*h);
TAC=Cu*D+Tc;
printf("Set up Cost (given) Co =%0.2f Rs per order\n",Co);
printf("Unit Cost (given) Cu=%0.2f Rs  per unit\n",Cu);
printf("Production rate (given) P=%0.2f per day\n",P);
printf("Inventory Carrying Cost Cc=%0.2f Rs/unit/unit time\n",Cc);
printf("Rate of interest(given) i=%0.3f per year\n\n",i);
printf("Economic Production quantity EPQ=%0.2f\n\n",EPQ);
printf("Number of Production runs per year N=%0.2f\n\n",N);
printf("Number of days between each production t=%0.2f\n\n",t);
printf("Total Annual inventory cost Tc=Rs %0.2f\n\n",Tc);
printf("Total Annual cost including cost material TAC=Rs %0.2f\n",TAC);
getch();
}
```

OUT PUT

```
enter the value of given demand for one week d
500
The annual demand D is=26000.00 per year
Enter the Ordering cost Co , Unit cost Cu and rate of interest i1 (interest perc
entage) according to given problem
75
50
7.5

Ordering Cost (given) Co =75.00 Rs per order
Unit Cost (given) Cu=50.00 Rs  per unit
Inventory Carrying Cost Cc=3.75 Rs/unit/unit time
Rate of interest i=0.075 per year

Economic order quantity EOQ=1019.80

Number of orders N=25.50

Number of days between each order t=14.32

Total inventory cost Tc=Rs 3824.26

Total cost including cost material TAC=Rs 1303824.25
```

PROGRAM 2–PRODUCTION MODEL

PROGRAM-3 FINITE RATE WITH SHORTAGE

```
/* TO SOLVE THE PROBLEM OF INVENTORY LEVEL MODEL3 finite rate
with shortage*/
#include<stdio.h>
#include<math.h>

void main()
{
float d1,d2,d,h,P,D,Cc,CC,Co,i,j,k,S,I,CS,Cs,Q,N,t,T,Tc,EPQ;
clrscr();
printf("enter the value of given demand per day d1 and daily demand(consumption
)rate d per day \n");
scanf("%f%f",&d1,&d2);
D=d1*365;
d=d2;
printf(" The annual demand D is=%0.2f per year\n",D);
printf("daily demand=%0.2f per day\n",d);
printf(" Enter the Production rate P per day  according to problem\n");
scanf("%f",&P);
printf("Enter the Set up(ordering) cost Co , Carrying cost(holding cost) Cc
/unit/day and shortage cost(stock out cost) Cs /unit/year according to given problem
\n");
scanf("%f%f%f",&Co,&CC,&CS);
Cs=CS*365;
Cc=CC*365;
h=1-(d/P);
k=((Cs+Cc)/Cs);
j=(Cs/(Cs+Cc)) ;
EPQ=sqrt((2*D*Co*k)/(Cc*h));
I=sqrt((2*D*Co*j)/(Cc*h));
S=EPQ-I;
N=EPQ/D;
t=365/N;
Tc=sqrt(2*D*Co*Cc*h*j);
printf("Set up Cost (given) Co =%0.2f Rs per order\n",Co);
printf("Inventory Carrying Cost Cc=%0.2f Rs/unit/unit year\n",Cc);
printf("Shortage (stock out cost) cost Cs=%0.2f Rs/unit/year\n\n",Cs);
printf("Economic Production quantity EPQ=%f\n\n",EPQ);
printf("Maximum Inventory level I =%0.2f\n",I);
printf("Production Period( Manufacturing Time) per year N=%f\n\n",N);
printf("Number of shortages=%0.2f\n",S);
printf("Number of days between each production t=%0.2f\n\n",t);
printf("Total Optimal Annual inventory cost Tc=Rs %0.2f\n\n",Tc);
getch();
}
```

OUTPUT

```
enter the value of given demand per month d1 and daily demand(consumption )rate
d per day
1000
30
 The annual demand D is=12000.00 per year
daily demand=30.00 per day
 Enter the Production rate P per day  according to problem
50

Enter the Set up cost Co , Unit cost Cu and rate of interest i1 (interest percen
tage) according to given problem
100
10
15

Set up Cost (given) Co =100.00 Rs per order
Unit Cost (given) Cu=10.00 Rs  per unit
Production rate (given) P=50.00 per day
Inventory Carrying Cost Cc=1.50 Rs/unit/unit time
Rate of interest(given) i=0.150 per year

Economic Production quantity EPQ=2000.00

Number of Production runs per year N=6.00

Number of days between each production t=60.83

Total Annual inventory cost Tc=Rs 1200.00

Total Annual cost including cost material TAC=Rs 121200.00
```

PROGRAM–3 FINITE RATE WITH SHORTAGE

PROGRAM-3 FINITE RATE WITH SHORTAGE

```
/* TO SOLVE THE PROBLEM OF INVENTORY LEVEL MODEL3 finite rate
with shortage*/
#include<stdio.h>
#include<math.h>

void main()
{
float d1,d2,d,h,P,D,Cc,CC,Co,i,j,k,S,I,CS,Cs,Q,N,t,T,Tc,EPQ;
clrscr();
printf("enter the value of given demand per day d1 and daily demand(consumption
)rate d per day \n");
scanf("%f%f",&d1,&d2);
D=d1*365;
d=d2;
printf(" The annual demand D is=%0.2f per year\n",D);
printf("daily demand=%0.2f per day\n",d);
printf(" Enter the Production rate P per day  according to problem\n");
scanf("%f",&P);
printf("Enter the Set up(ordering) cost Co , Carrying cost(holding cost) Cc
/unit/day and shortage cost(stock out cost) Cs /unit/year according to given problem
\n");
scanf("%f%f%f",&Co,&CC,&CS);
Cs=CS*365;
Cc=CC*365;
h=1-(d/P);
k=((Cs+Cc)/Cs);
j=(Cs/(Cs+Cc)) ;
EPQ=sqrt((2*D*Co*k)/(Cc*h));
I=sqrt((2*D*Co*j)/(Cc*h));
S=EPQ-I;
N=EPQ/D;
t=365/N;
Tc=sqrt(2*D*Co*Cc*h*j);
printf("Set up Cost (given) Co =%0.2f Rs per order\n",Co);
printf("Inventory Carrying Cost Cc=%0.2f Rs/unit/unit year\n",Cc);
printf("Shortage (stock out cost) cost Cs=%0.2f Rs/unit/year\n\n",Cs);
printf("Economic Production quantity EPQ=%f\n\n",EPQ);
printf("Maximum Inventory level I =%0.2f\n",I);
printf("Production Period( Manufacturing Time) per year N=%f\n\n",N);
printf("Number of shortages=%0.2f\n",S);
printf("Number of days between each production t=%0.2f\n\n",t);
printf("Total Optimal Annual inventory cost Tc=Rs %0.2f\n\n",Tc);
getch();
}
```

OUTPUT:

```
enter the value of given demand per day d1 and daily demand(consumption )rate d
per day
200
200
 The annual demand D is=73000.00 per year
daily demand=200.00 per day
 Enter the Production rate P per day  according to problem
400
Enter the Set up(ordering) cost Co , Carrying cost(holding cost) Cc /unit/day an
d shortage cost(stock out cost) Cs /unit/year according to given problem
50
2
10
Set up Cost (given) Co =50.00 Rs per order
Inventory Carrying Cost Cc=730.00 Rs/unit/unit year
Shortage (stock out cost) cost Cs=3650.00 Rs/unit/year

Economic Production quantity EPQ=154.919342

Maximum Inventory level I =129.10
Production Period( Manufacturing Time) per year N=0.002122

Number of shortages=25.82
Number of days between each production t=171992.72

Total Optimal Annual inventory cost Tc=Rs 47121.30
```

PROGRAM–4 INVENTORY DISCOUNT MODEL

PROGRAM 4– INVENTORY DISCOUNT MODEL

```
/* To solve the Inventory Discount Model Problem*/
#include<stdio.h>
#include<math.h>
void main()
{
float D,Cu1,Cu2,Cu3,Co,Q1,Q2,Q3,EOQ,EOQ1,EOQ2,EOQ3;
float H1,K1,H2,K2,H3,K3,Q11,TAC,TAC1,TAC2,TAC3,i,i1;
clrscr();
printf("Enter the value of annual demand D per year and Ordering Cost Co \n and
inventory carrying cost percentage i\n");
scanf("%f%f%f",&D,&Co,&i);
printf(" Enter the value of Unit costs Cu1,Cu2,Cu3 in Rs \n");
scanf("%f%f%f",&Cu1,&Cu2,&Cu3);
printf(" Enter the value of Quantities Q1,Q2 AND Q3\n");
scanf("%f%f%f",&Q1,&Q2,&Q3);
printf("      QUANTITY              PRICE/UNIT (Rs) \n");
printf("  %0.2f <= Q <= %0.2f,        Cu1=%0.2f Rs\n",Q1,Q2,Cu1);
printf("  %0.2f <= Q <= %0.2f,        Cu2=%0.2f Rs\n",Q2,Q3,Cu2);
printf("       Q >= %0.2f,            Cu3=%0.2f Rs\n",Q3,Cu3);
i1=i/100;
printf("i1=%f\n",i1);
EOQ1=sqrt((2*D*Co)/(Cu1*i1));
EOQ2=sqrt((2*D*Co)/(Cu2*i1));
EOQ3=sqrt((2*D*Co)/(Cu3*i1));
printf("Economic Order Quantity EOQ1=%0.2f\n",EOQ1);
printf("Economic Order Quantity EOQ2=%0.2f\n",EOQ2);
printf("Economic Order Quantity EOQ3=%0.2f\n",EOQ3);
if( EOQ2 < EOQ1 && EOQ2 < EOQ3)
{
printf(" EOQ2 is the smallest value which represents the best quantity in that
volume range=%f\n",EOQ2);
printf(" Determine quantity to be purchased at each price level are\n");
printf(" EOQ2=%0.2f\n",EOQ2);
Q11=EOQ2;
printf(" Selected Quantity Q11=%0.2f\n",Q11);
printf(" Selected Quantity Q2=%0.2f\n",Q2);
printf(" Selected Quantity Q3=%0.2f\n",Q3);
H1=D/Q11;
K1=Q11/2;
TAC1=((Cu1*D)+(H1*Co)+(K1*Cu1*i1));
H2=D/Q2;
K2=Q2/2;
TAC2=((Cu2*D)+(H2*Co)+(K2*Cu2*i1));
H3=D/Q3;
```

```
K3=Q3/2;
TAC3=((Cu3*D)+(H3*Co)+(K3*Cu3*i1));
printf(" Total annual cost including materials for all selected quantities
TAC(Q1)=%0.3f Rs \n",TAC1);
printf(" Total annual cost including materials for all selected quantities
TAC(Q2)=%0.3f Rs \n",TAC2);
printf(" Total annual cost including materials for all selected quantities
TAC(Q3)=%0.3f Rs \n",TAC3);
}
else if ( EOQ1 < EOQ2 && EOQ1 < EOQ3)
{
printf(" EOQ1 is the smallest value which represents the best quantity in that
volume range=%f\n",EOQ1);
printf(" Determine quantity to be purchased at each price level are\n");
printf(" EOQ1=%0.2f\n",EOQ1);
Q11=EOQ1;
printf("Selected Quantity Q11=%0.2f\n",Q11);
printf("Selected Quantity Q2=%0.2f\n",Q2);
printf("Selected Quantity Q3=%0.2f\n",Q3);
H1=D/Q11;
K1=Q11/2;
TAC1=((Cu1*D)+(H1*Co)+(K1*Cu1*i1));
H2=D/Q2;
K2=Q2/2;
TAC2=((Cu2*D)+(H2*Co)+(K2*Cu2*i1));
H3=D/Q3;
K3=Q3/2;
TAC3=((Cu3*D)+(H3*Co)+(K3*Cu3*i1));
printf(" Total annual cost including materials for all selected quantities
TAC(Q1)=%0.3f Rs\n",TAC1);
printf(" Total annual cost including materials for all selected quantities
TAC(Q2)=%0.3f Rs\n",TAC2);
printf(" Total annual cost including materials for all selected quantities
TAC(Q3)=%0.3f Rs\n",TAC3);
}
else
{
printf(" EOQ3 is the smallest value which represents the best quantity in that
volume range =%f\n",EOQ3);
printf(" Determine quantity to be purchased at each price level are\n");
printf(" EOQ3=%0.2f\n",EOQ3);
Q11=EOQ3;
printf("Selected Quantity Q11=%0.2f\n",Q11);
printf("Selected Quantity Q2=%0.2f\n",Q2);
printf("Selected Quantity Q3=%0.2f\n",Q3);
H1=D/Q11;
```

```
K1=Q11/2;
TAC1=((Cu1*D)+(H1*Co)+(K1*Cu1*i1));
H2=D/Q2;
K2=Q2/2;
TAC2=((Cu2*D)+(H2*Co)+(K2*Cu2*i1));
H3=D/Q3;
K3=Q3/2;
TAC3=((Cu3*D)+(H3*Co)+(K3*Cu3*i1));
printf(" Total annual cost including materials for all selected quantities
TAC(Q1)=%0.3f Rs\n",TAC1);
printf(" Total annual cost including materials for all selected quantities
TAC(Q2)=%0.3f Rs\n",TAC2);
printf(" Total annual cost including materials for all selected quantities
TAC(Q3)=%0.3f Rs\n",TAC3);
}
if( TAC2 < TAC1 && TAC2 < TAC3)
{
printf(" The Optimal lowest annual cost TAC2 =%0.2f Rs\n", TAC2);
}
else if (TAC1 < TAC2 && TAC1 < TAC3)
{
printf(" The Optimal lowest annual cost TAC1 =%0.2f Rs\n", TAC1);
}
else
{
printf(" The Optimal lowest annual cost TAC3 =%0.2f Rs\n", TAC3);
}
getch();
}
```

OUTPUT:

```
Enter the value of annual demand D per year and Ordering Cost Co
 and inventory carrying cost percentage i
5400
600
30
 Enter the value of Unit costs Cu1,Cu2,Cu3 in Rs
12
10
8
 Enter the value of Quantities Q1,Q2 AND Q3
0
2400
3000
      QUANTITY                  PRICE/UNIT (Rs)
  0.00 <= Q <= 2400.00,          Cu1=12.00 Rs
  2400.00 <= Q <= 3000.00,       Cu2=10.00 Rs
        Q >= 3000.00,            Cu3=8.00 Rs
i1=0.300000
Economic Order Quantity EOQ1=1341.64
Economic Order Quantity EOQ2=1469.69
Economic Order Quantity EOQ3=1643.17
 EOQ1 is the smallest value which represents the best quantity in that volume ra
nge=1341.640747
 Determine quantity to be purchased at each price level are
 EOQ1=1341.64
Selected Quantity Q11=1341.64
Selected Quantity Q2=2400.00
Selected Quantity Q3=3000.00
 Total annual cost including materials for all selected quantities TAC(Q1)=69629
.906 Rs
 Total annual cost including materials for all selected quantities TAC(Q2)=58950
.000 Rs
 Total annual cost including materials for all selected quantities TAC(Q3)=47880
.000 Rs
 The Optimal lowest annual cost TAC3 =47880.00 Rs
```

QUEUEING THEORY in "C"

QUEUEING THEORY includes

(1) Input (arrival pattern)
(2) The Service mechanism (service pattern)
(3) Queue Discipline
(4) Customer's behaviour

The whole description of queueing theory is given in our book.

Here we are giving a two C–programs which include the properties of MODEL–I, FCFS.

In First program, you should put the values of arrival rate and service rate in minutes. And then these values will convert mean arrival and service rate in per minute. You can know about the many properties of first come first serve model.

In second program, you enter the values of mean arrival rate and service rate in per hour directly and then you find the properties of first model.

Similarly you can make many programs of queueing theory models. In the next edition of the book, we will provide other programs.

Your suggestions or comments is very necessary for modifying this Program.

—AUTHOR

PROGRAM–1

PROGRAM-1

/* To find the properties of Queueing theory (MODEL-1 (M|M|1):(...|FCFS))*/

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

void main()
{
float a1,u1,a,u,p,Ls,Lq,Wq,Ws,WW,LL,h,k,QL,P,N,L,P1,P2;
clrscr();
printf("              Q U E U E I N G   T H E O R Y        \n");
printf("Queueing theory problems depends on arrival rate and service rate
values\n");
printf("Both the value of arrival & service rate u should be in per unit time (seconds
,minutes or hours)\n");
printf("For example,if arrival rate and service rate value is given in minutes,we
should convert the value in per minute\n");
printf("If one of or both the values of arrival rate and service rate is given in hours
or days,we should convert both the values in per hour or per day\n\n");
printf("Enter the value of arrival rate a1 and service rate u1 in minutes\n");
scanf("%f%f",&a1,&u1);
a=1/a1;
u=1/u1;
printf("Mean arrival rate (a=1/a1 per minute) a =%f\n",a);
printf("Mean service rate (u=1/u1 per minute) u =%f\n",u);
p=a/u;
Ls=p/(1-p);
Lq=Ls-p;
Wq=p/(u*(1-p));
Ws=1/(u*(1-p));
WW=1/(u*(1-p));
LL=1/(1-p);
h=pow(p,2);
QL=p/(1-h);
P1=1-p;
P2=1-(1-p);
printf("Traffic intensity (or utilization factor) for service facility p=%f\n\n",p);
printf("Expected no of customers in the system or expected line length Ls=%0.2f
customers\n",Ls);
printf("Expected No of customers in Queue or expected queue length Lq=%0.2f
customers\n",Lq);
printf("Expected waiting time per customer in the Queue ( excluding service time)\n
Wq=%0.2f minutes\n",Wq);
printf("Expected waiting time per customer in the system( including service time)\n
Ws=%0.2f minutes\n",Ws);
```

```
printf("Expected waiting time of a customer who has to wait,(W|W > 0)=
%0.2f\n",WW);
printf("Expected length of Non empty Queue ,( L|L > 0)=%0.2f\n",LL);
printf("Queue length is QL=%f\n",QL);
printf("the proportion  of time, the server is idle P1=%f\n",P1);
printf(" the proportion of time ,the server is busy P2=%f\n",P2);
printf("To find the probability for Queue size , enter the value of Queue size N (like
1,2,3,4.....)\n");
scanf("%f",&N);
k=pow(p,N);
P=k;
printf("The probability when queue size is exceeding in the system Prob[ queue size
>= %0.2f] =%0.3f\n",N,P);
getch();
}
```

OUTPUT:

```
                QUEUEING  THEORY
Queueing theory problems depends on arrival rate and service rate values
Both the value of arrival & service rate u should be in per unit time (seconds ,
minutes or hours)
For example,if arrival rate and service rate value is given in minutes,we should
 convert the value in per minute
If one of or both the values of arrival rate and service rate is given in hours
or days,we should convert both the values in per hour or per day

Enter the value of arrival rate a1 and service rate u1 in minutes
48
36

Mean arrival rate (a=1/a1 per minute) a =0.020833
Mean service rate (u=1/u1 per minute) u =0.027778
Traffic intensity (or utilization factor) for service facility p=0.750000

Expected no of customers in the system or expected line length Ls=3.00 customers

Expected No of customers in Queue or expected queue length Lq=2.25 customers
Expected waiting time per customer in the Queue ( excluding service time)
 Wq=108.00 minutes
Expected waiting time per customer in the system( including service time)
 Ws=144.00 minutes
Expected waiting time of a customer who has to wait,(W|W > 0)= 144.00
Expected length of Non empty Queue ,( L|L > 0)=4.00
Queue length is QL=1.714286
the proportion  of time, the server is idle P1=0.250000
```

the proportion of time ,the server is busy P2=0.750000
To find the probability for Queue size , enter the value of Queue size N (like 1 ,2,3,4.....)

10
The probability when queue size is exceeding in the system Prob[queue size >= 10.00] =0.056

PROGRAM–2

Program-2

```
/* To find the properties of Queueing theory (MODEL-1 (M|M|1):(...|FCFS))*/
#include<stdio.h>
#include<math.h>

void main()
{
float a1,u1,a,u,p,Ls,Lq,Wq,Ws,WW,LL,h,k,QL,P,N,L,P1,P2;
clrscr();
printf("              QUEUEING  THEORY        \n");
printf("Queueing theory problems depends on arrival rate and service rate
values\n");
printf("Both the value of arrival a & service rate u should be in per unit time
(seconds ,minutes or hours)\n");
printf("For example,if arrival rate and service rate value is given in minutes,we
should convert the value in per minute\n");
printf("If one of or both the values of arrival rate and service rate is given in hours
or days,we should convert both the values in per hour or per day\n\n");
printf("Enter the value of mean arrival rate a and service rate u in per hour\n");
scanf("%f%f",&a,&u);
p=a/u;
Ls=p/(1-p);
Lq=Ls-p;
Wq=p/(u*(1-p));
Ws=1/(u*(1-p));
WW=1/(u*(1-p));
LL=1/(1-p);
h=pow(p,2);
QL=p/(1-h);
P1=1-p;
P2=1-(1-p);
printf("Traffic intensity (or utilization factor) for service facility p=%f\n\n",p);
printf("Expected no of customers in the system or expected line length Ls=%0.2f
customers\n",Ls);
printf("Expected No of customers in Queue or expected queue length Lq=%0.2f
customers\n",Lq);
printf("Expected waiting time per customer in the Queue ( excluding service time)\n
Wq=%0.2f hr\n",Wq);
printf("Expected waiting time per customer in the system( including service time)\n
Ws=%0.2f hr\n",Ws);
printf("Expected waiting time of a customer who has to wait,(W|W > 0)=
%0.2f\n",WW);
printf("Expected length of Non empty Queue ,( L|L > 0)=%0.2f\n",LL);
printf("Queue length is QL=%f\n",QL);
printf("the proportion  of time, the server is idle P1=%f\n",P1);
```

```
printf(" the proportion of time ,the server is busy P2=%f\n",P2);
printf("To find the probability for Queue size , enter the value of Queue size N (like
1,2,3,4.....)\n");
scanf("%f",&N);
k=pow(p,N);
P=k;
printf("The probability when queue size is exceeding in the system Prob[ queue size
>= %0.2f] =%0.3f\n",N,P);
getch();
}
```

OUTPUT:

```
                QUEUEING  THEORY
Queueing theory problems depends on arrival rate and service rate values
Both the values of arrival a & service rate u should be in per unit time (seconds
 ,minutes or hours)
For example,if arrival rate and service rate value is given in minutes,we should
 convert the value in per minute
If one of or both the values of arrival rate and service rate is given in hours
or days,we should convert both the values in per hour or per day

Enter the value of mean arrival rate a and service rate u in per hour
8
12

Traffic intensity (or utilization factor) for service facility p=0.666667

Expected no of customers in the system or expected line length Ls=2.00 customers

Expected No of customers in Queue or expected queue length Lq=1.33 customers
Expected waiting time per customer in the Queue ( excluding service time)
 Wq=0.17 hr
Expected waiting time per customer in the system( including service time)
 Ws=0.25 hr
Expected waiting time of a customer who has to wait,(W|W > 0)= 0.25
Expected length of Non empty Queue ,( L|L > 0)=3.00
Queue length is QL=1.200000
the proportion  of time, the server is idle P1=0.333333
 the proportion of time ,the server is busy P2=0.666667
To find the probability for Queue size , enter the value of Queue size N (like 1
,2,3,4.....)
2
The probability when queue size is exceeding in the system Prob[ queue size >= 2
.00] =0.444
```

■■■

APPENDIX – A

A New Method For Initial Solution Of Transportation Problem

A-1 MATHEMATICAL FORMULATION OF T.P.

There are several methods for finding the *initial basic feasible solution* of Transportation Problem (T.P.) as discussed in **Chapter 13.** But, there is no suitable answer to the question : which method is the beast one ? In this article, we have developed a new technique for finding the nearly optimal solution which requires less iterations to reach optimality in comparison to the methods available in the literature. The degeneracy problem is also avoided by this method. This method is better than *Vogel's Approximation Method* also, and is based on the min-max (max-min) criteria of *'Game Theory'*

Mathematical formulation of Transportation Problem has been given in **Chapter 10.**

A-2 MIN(MIN-MAX) ALGORITHM*

Step 1. Choose the maximum cost (c_{ij}) cell ($i = 1, ..., m$; $j = 1, ..., n$). In case of ties, choose the maximum arbitrarily.

Step 2. (*i*) Choose the minimum cost cell in the row containing the maximum (c_{ij}).

(*ii*) Allocate the maximum possible quantity to the minimum cost cell.

(*iii*) Find the total cost of this allocation.

Step 3. Repeat *Step* 2 for the column cell containing the maximum (c_{ij}).

Step 4. Choose for allocation the cell with minimum cost. In case of a tie choose arbitrarily.

Step 5. Delete the row (column) when the allocation becomes complete. In case of a tie delete either row or column. If the row and column both deserve to be deleted, put zero in the minimum cost cell of either row of column which are not yet deleted.

Illustrative Example

Example : Consider the problem :

19	30	50	10	7
70	30	40	60	9
40	8	70	20	18
5	8	7	14	

Proceeding step-by-step according to the algorithm suggested above, we get the following solution :

19 **5**	30	50	10 **2**	7
70	30	40 **7**	60 **2**	9
40	8 **8**	70	20 **10**	18
5	8	7	14	

The initial cost = 779 units.

*This method was developed by the author in 1989 when he was working on deputation as Professor of Operations Research in the University of Salahaddin, Arbil, Iraq. On this article the author was awarded by the *"Ministry of Higher Education and Scientific Research (Iraq).*

However, if we find the initial solution of this problem by *Vogel's Method* we reach the same solution. But this new approach requires less computations as well as the total number of basic cells are always $m+n-1$, thus avoiding the problem of degeneracy.

A-3 MAX (MIN-MAX) ALGORITHM

Step 1. Choose the max (c_{ij}) cell $(i = 1, ..., m\ ; j = 1, ..., n)$. In case of ties, choose the maximum arbitrarily.

Step 2. (*i*) Choose the minimum cost cellin the now containing the maximum (c_{ij})

(*ii*) Allocate the maximum possible quantity to the minimum cost cell.

(*iii*) Find the total cost of this allocation.

Step 3. Repeat *Step 2* for the column cell containing the maximum (c_{ij}).

Step 4. Choose for allocation the cell with maximum total cost. In case of a tie choose arbitrarily.

Step 5. Delete the row (column) when the allocation becomes complete. In case of tie delete either row or column. If the row and column both deserve do be delete, put zero in the minimum cost cell which are not yet deleted.

Step 6. When all $m+n-1$ cells are allocated, stop. Otherwise, go to *Step* 1.

A-4 VERIFICATION BY EXAMPLE

Following example verifies that the initial solution becomes optimal when

max (min – max) = min (min – max).

Conisder the following initial BFS obtained by both the above algorithms :

4 2	1 3	4	2	5
5	2 6	5	3	6
6	3	1 5	1 2	7
2	9	5	2	

Max		Min	Max
6	Row	$1 \times 5 = 5$	
	Column	$4 \times 2 = 8$*	a_{11}, Omit column 1
5	Row	$2 \times 6 = 12$*	a_{22}, Omit Row 2
	Column	$1 \times 5 = 5$	
3	Row	$1 \times 2 = 2$	
	Column	1×3*	a_{12}, Omit Row 1

Note : To avoid degeneracy we allocate empty (zero) to the cell a_{32} and we omit column 2.

1 Row and Column $1 \times 2 = 2$

which is the last cell solution. Total cost is 30 and this solution satisfies the test of optimality.

A-5 CONCLUDING REMARK

The algorithm developed in this article has the following major advantages :

(*i*) It requires less computations in comparison to the methods existing in the literature.

(*ii*) The problem of degeneracy will not arise in the initial BFS.

(*iii*) The algorithm has the ***interesting property*** : max (min - max) = min (min - max) = optimum solution.

✹✹✹