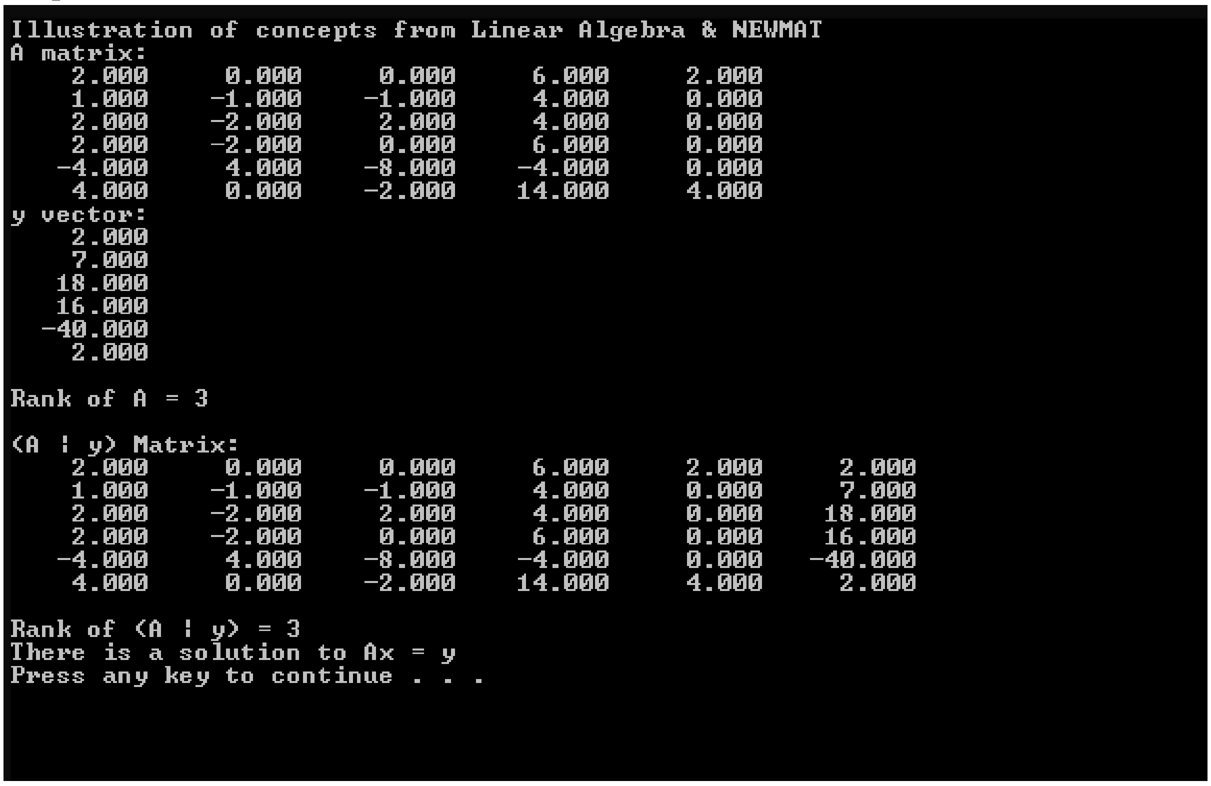
IE531: Algorithms for Data Analytics Spring, 2018

Karthik Venkata(kvn3)

Homework 1: Review of Linear Algebra, Probability & Statistics and Computing

1. **(a).** Upon modifying and running the Linear Algebra 1.cpp program, the output obtained was the follows:



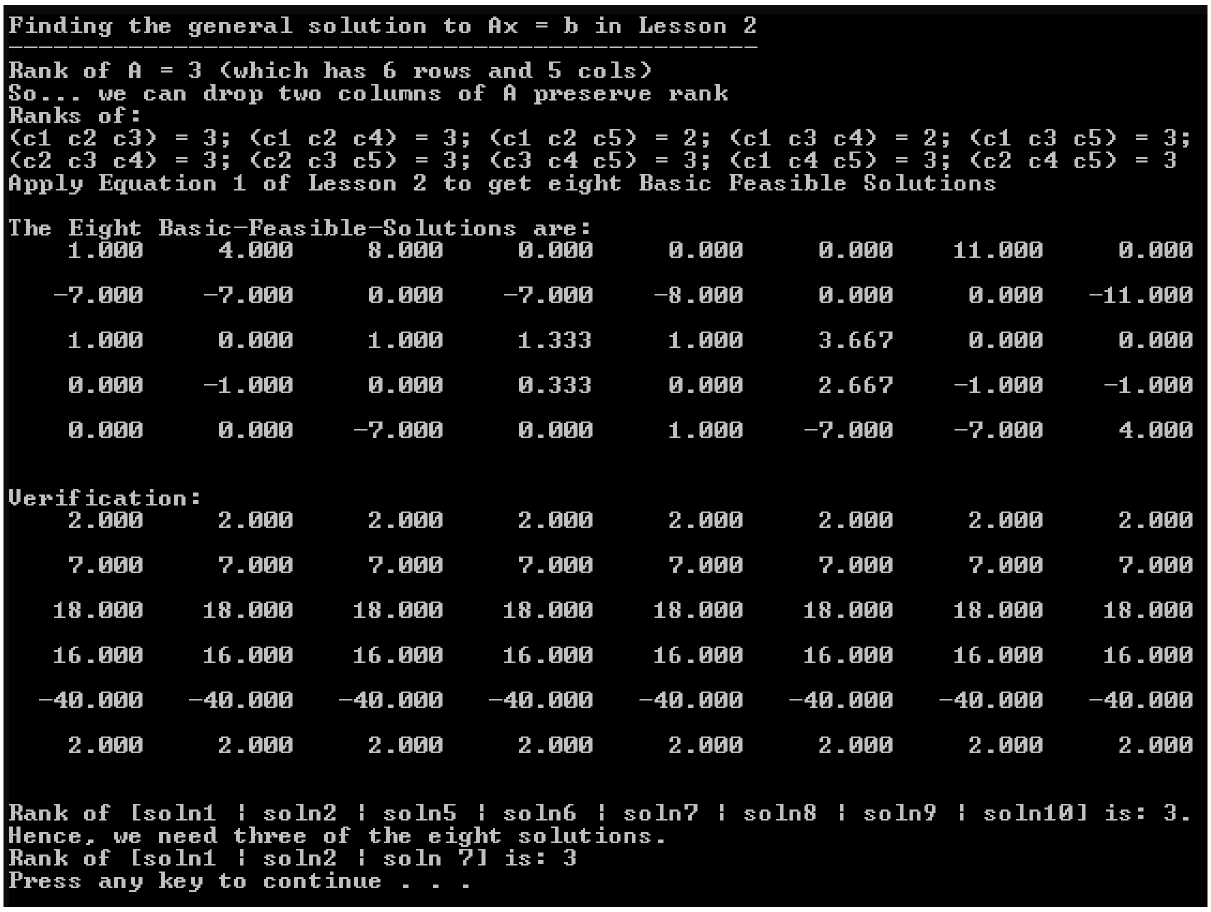
```
Illustration of concepts from Linear Algebra & NEWMAT
A matrix:
    2.000     0.000     0.000     6.000     2.000
    1.000    -1.000    -1.000     4.000     0.000
    2.000    -2.000     2.000     4.000     0.000
    2.000    -2.000     0.000     6.000     0.000
   -4.000     4.000    -8.000    -4.000     0.000
    4.000     0.000    -2.000    14.000     4.000
y vector:
    2.000
    7.000
   18.000
   16.000
  -40.000
    2.000

Rank of A = 3

(A | y) Matrix:
    2.000     0.000     0.000     6.000     2.000     2.000
    1.000    -1.000    -1.000     4.000     0.000     7.000
    2.000    -2.000     2.000     4.000     0.000    18.000
    2.000    -2.000     0.000     6.000     0.000    16.000
   -4.000     4.000    -8.000    -4.000     0.000   -40.000
    4.000     0.000    -2.000    14.000     4.000     2.000

Rank of (A | y) = 3
There is a solution to Ax = y
Press any key to continue . . .
```

1. **(b)** The output generated upon running the modified version of the Linear_Algebra_2.cpp for computing the solutions of the given set of equations is-



```
Finding the general solution to Ax = b in Lesson 2
--------------------------------------------------
Rank of A = 3 (which has 6 rows and 5 cols)
So... we can drop two columns of A preserve rank
Ranks of:
(c1 c2 c3) = 3; (c1 c2 c4) = 3; (c1 c2 c5) = 2; (c1 c3 c4) = 2; (c1 c3 c5) = 3;
(c2 c3 c4) = 3; (c2 c3 c5) = 3; (c3 c4 c5) = 3; (c1 c4 c5) = 3; (c2 c4 c5) = 3
Apply Equation 1 of Lesson 2 to get eight Basic Feasible Solutions

The Eight Basic-Feasible-Solutions are:
    1.000     4.000     8.000     0.000     0.000     0.000    11.000     0.000

   -7.000    -7.000     0.000    -7.000    -8.000     0.000     0.000   -11.000

    1.000     0.000     1.000     1.333     1.000     3.667     0.000     0.000

    0.000    -1.000     0.000     0.333     0.000     2.667    -1.000    -1.000

    0.000     0.000    -7.000     0.000     1.000    -7.000    -7.000     4.000


Verification:
    2.000     2.000     2.000     2.000     2.000     2.000     2.000     2.000

    7.000     7.000     7.000     7.000     7.000     7.000     7.000     7.000

   18.000    18.000    18.000    18.000    18.000    18.000    18.000    18.000

   16.000    16.000    16.000    16.000    16.000    16.000    16.000    16.000

  -40.000   -40.000   -40.000   -40.000   -40.000   -40.000   -40.000   -40.000

    2.000     2.000     2.000     2.000     2.000     2.000     2.000     2.000


Rank of [soln1 | soln2 | soln5 | soln6 | soln7 | soln8 | soln9 | soln10] is: 3.
Hence, we need three of the eight solutions.
Rank of [soln1 | soln2 | soln 7] is: 3
Press any key to continue . . .
```

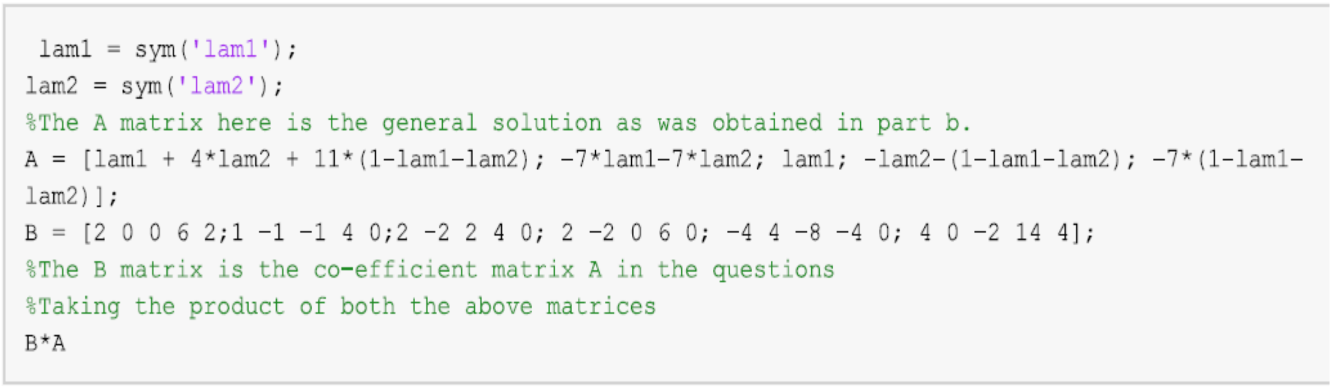
Therefore, we find that the rank of the set of column (solution) vectors 1, 2 and 7 is 3 and the rank of the entire set of solutions (column vectors) is also 3. Hence, the basis set which gives the complete solution set consists of columns 1, 2 and 7. The general solution to these set of equations (an affine combination of all the solutions as obtained above) is as follows-

$$\begin{bmatrix} 1 & 4 & 11 \\ -7 & -7 & 0 \\ 1 & 0 & 0 \\ 0 & -1 & -1 \\ 0 & 0 & -7 \end{bmatrix}$$

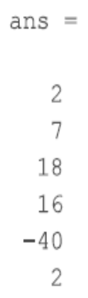
The general solution to these set of equations is of the form-

$$\begin{bmatrix} \lambda1 + 4\lambda2 + 11(1 - \lambda1 - \lambda2) \\ -7\,\lambda1 - 7\,\lambda2 \\ \lambda1 \\ -\lambda2 - (1 - \lambda1 - \lambda2) \\ -7(1 - \lambda1 - \lambda2) \end{bmatrix}$$

1.(c). The general formula obtained in part (b) was plugged into the equation **Ax = y** and was verified using MATLAB. The MATLAB snapshot of the same is as below:



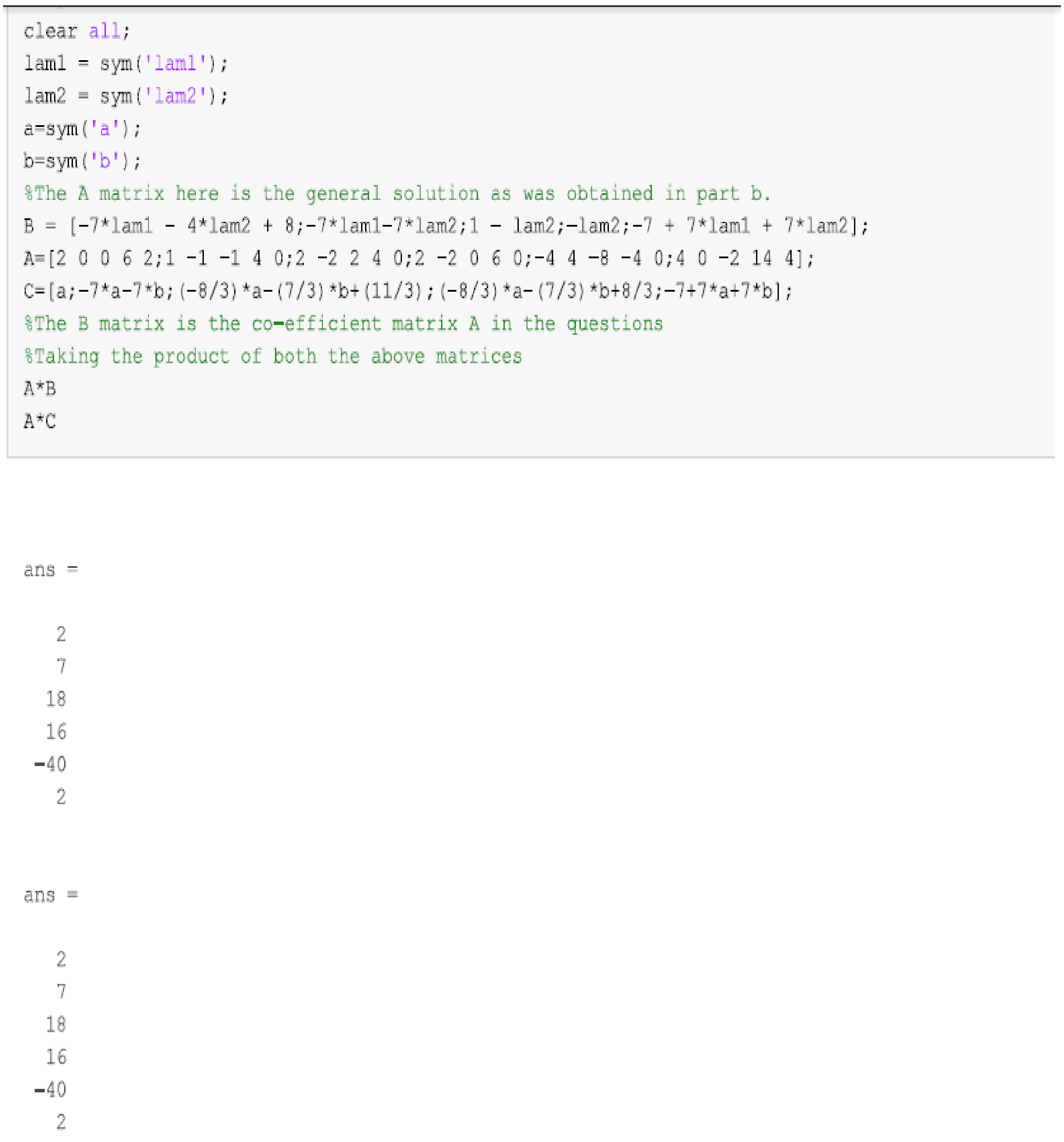
```
 lam1 = sym('lam1');
lam2 = sym('lam2');
%The A matrix here is the general solution as was obtained in part b.
A = [lam1 + 4*lam2 + 11*(1-lam1-lam2); -7*lam1-7*lam2; lam1; -lam2-(1-lam1-lam2); -7*(1-lam1-
lam2)];
B = [2 0 0 6 2;1 -1 -1 4 0;2 -2 2 4 0; 2 -2 0 6 0; -4 4 -8 -4 0; 4 0 -2 14 4];
%The B matrix is the co-efficient matrix A in the questions
%Taking the product of both the above matrices
B*A
```



```
ans =

   2
   7
  18
  16
 -40
   2
```

1.(d). Both the given set of solutions were verified by plugging into the general equation **AX = y** and the following MATLAB snapshot shows the same:



```
clear all;
lam1 = sym('lam1');
lam2 = sym('lam2');
a=sym('a');
b=sym('b');
%The A matrix here is the general solution as was obtained in part b.
B = [-7*lam1 - 4*lam2 + 8;-7*lam1-7*lam2;1 - lam2;-lam2;-7 + 7*lam1 + 7*lam2];
A=[2 0 0 6 2;1 -1 -1 4 0;2 -2 2 4 0;2 -2 0 6 0;-4 4 -8 -4 0;4 0 -2 14 4];
C=[a;-7*a-7*b;(-8/3)*a-(7/3)*b+(11/3);(-8/3)*a-(7/3)*b+8/3;-7+7*a+7*b];
%The B matrix is the co-efficient matrix A in the questions
%Taking the product of both the above matrices
A*B
A*C
```

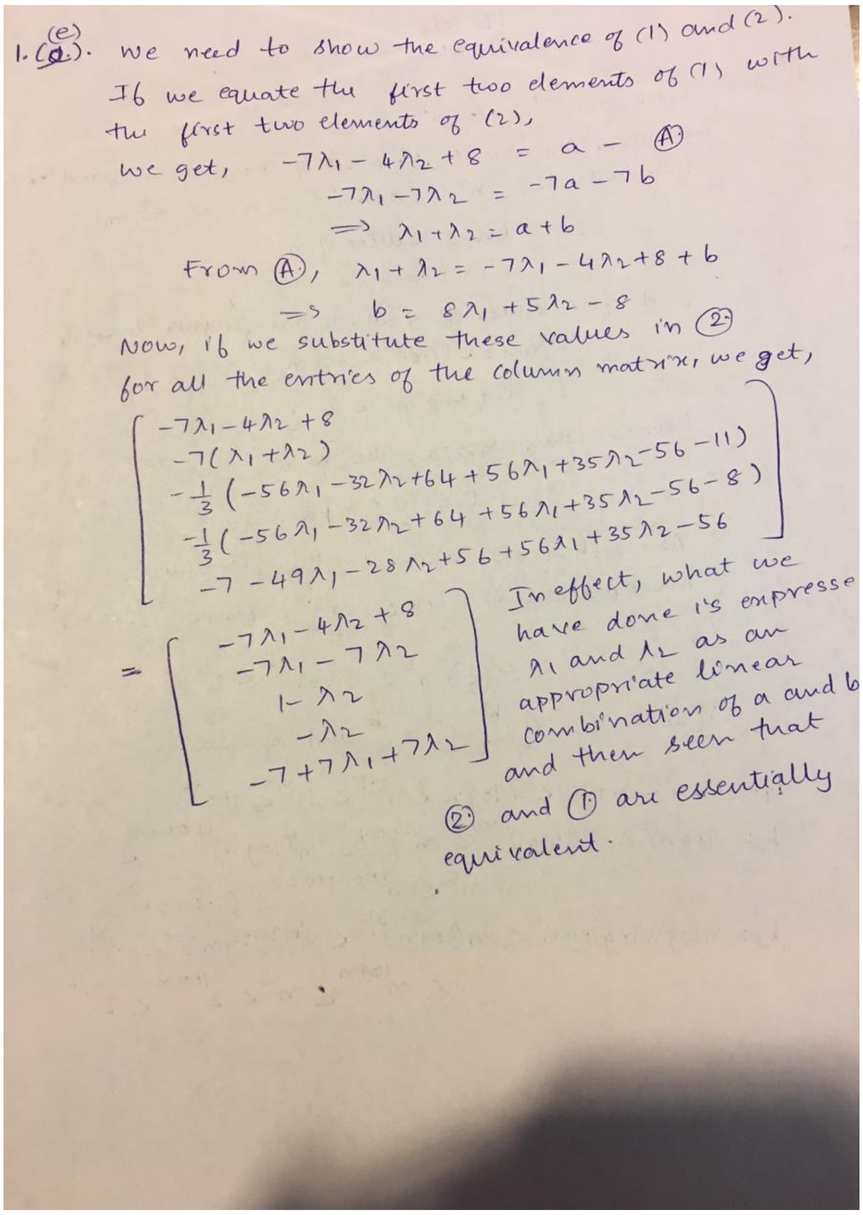
```
ans =

   2
   7
  18
  16
 -40
   2


ans =

   2
   7
  18
  16
 -40
   2
```

Correction-‘A’ matrix is the co-efficient matrix that is given in the question. ‘B’ and ‘C’ are the equivalent set of solution matrices that have been given.

(e). 

1. ~~(a)~~ (e). we need to show the equivalence of (1) and (2).

If we equate the first two elements of (1) with the first two elements of (2),

we get, $-7\lambda_1 - 4\lambda_2 + 8 = a$ — $\textcircled{A}$

$-7\lambda_1 - 7\lambda_2 = -7a - 7b$

$\Rightarrow \lambda_1 + \lambda_2 = a + b$

From $\textcircled{A}$, $\lambda_1 + \lambda_2 = -7\lambda_1 - 4\lambda_2 + 8 + b$

$\Rightarrow b = 8\lambda_1 + 5\lambda_2 - 8$

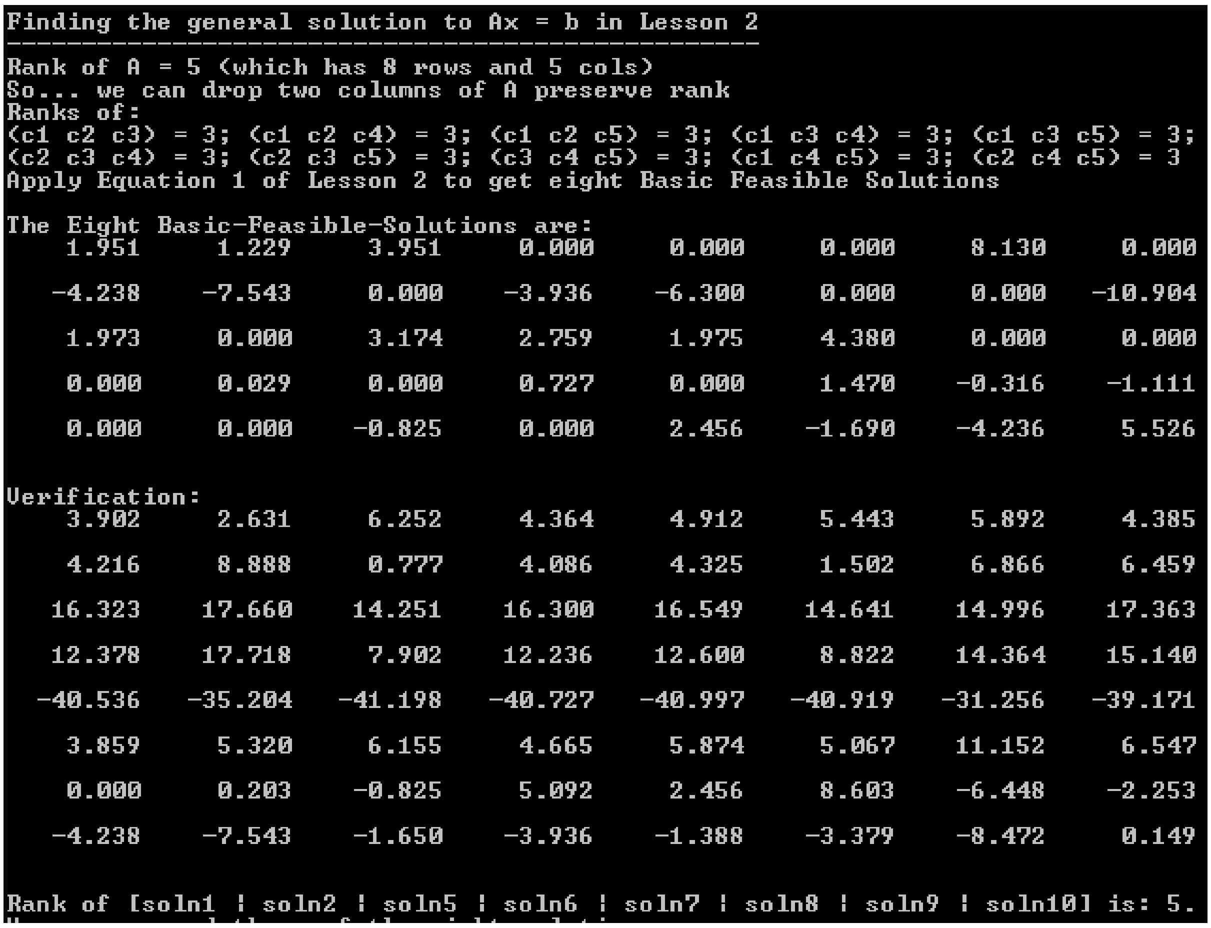
Now, if we substitute these values in $\textcircled{2}$ for all the entries of the column matrix, we get,

$$\begin{bmatrix} -7\lambda_1 - 4\lambda_2 + 8 \\ -7(\lambda_1 + \lambda_2) \\ -\frac{1}{3}(-56\lambda_1 - 32\lambda_2 + 64 + 56\lambda_1 + 35\lambda_2 - 56 - 11) \\ -\frac{1}{3}(-56\lambda_1 - 32\lambda_2 + 64 + 56\lambda_1 + 35\lambda_2 - 56 - 8) \\ -\frac{1}{3}(-56\lambda_1 - 32\lambda_2 + 64 + 56\lambda_1 + 35\lambda_2 - 56 \\ -7 - 49\lambda_1 - 28\lambda_2 + 56 + 56\lambda_1 + 35\lambda_2 - 56 \end{bmatrix}$$

$$= \begin{bmatrix} -7\lambda_1 - 4\lambda_2 + 8 \\ -7\lambda_1 - 7\lambda_2 \\ 1 - \lambda_2 \\ -\lambda_2 \\ -7 + 7\lambda_1 + 7\lambda_2 \end{bmatrix}$$

In effect, what we have done is expresse λ_1 and λ_2 as an appropriate linear combination of a and b and then seen that $\textcircled{2}$ and $\textcircled{1}$ are essentially equivalent.

(f). We see that as the rank of the A matrix is five which equals the number of columns. Hence, the given set of equations must have a unique solution as the coefficient matrix A is full rank. The output obtained from visual studio is as follows-



```
Finding the general solution to Ax = b in Lesson 2
--------------------------------------------------
Rank of A = 5 (which has 8 rows and 5 cols)
So... we can drop two columns of A preserve rank
Ranks of:
(c1 c2 c3) = 3; (c1 c2 c4) = 3; (c1 c2 c5) = 3; (c1 c3 c4) = 3; (c1 c3 c5) = 3;
(c2 c3 c4) = 3; (c2 c3 c5) = 3; (c3 c4 c5) = 3; (c1 c4 c5) = 3; (c2 c4 c5) = 3
Apply Equation 1 of Lesson 2 to get eight Basic Feasible Solutions

The Eight Basic-Feasible-Solutions are:
    1.951     1.229     3.951     0.000     0.000     0.000     8.130     0.000

   -4.238    -7.543     0.000    -3.936    -6.300     0.000     0.000   -10.904

    1.973     0.000     3.174     2.759     1.975     4.380     0.000     0.000

    0.000     0.029     0.000     0.727     0.000     1.470    -0.316    -1.111

    0.000     0.000    -0.825     0.000     2.456    -1.690    -4.236     5.526


Verification:
    3.902     2.631     6.252     4.364     4.912     5.443     5.892     4.385

    4.216     8.888     0.777     4.086     4.325     1.502     6.866     6.459

   16.323    17.660    14.251    16.300    16.549    14.641    14.996    17.363

   12.378    17.718     7.902    12.236    12.600     8.822    14.364    15.140

  -40.536   -35.204   -41.198   -40.727   -40.997   -40.919   -31.256   -39.171

    3.859     5.320     6.155     4.665     5.874     5.067    11.152     6.547

    0.000     0.203    -0.825     5.092     2.456     8.603    -6.448    -2.253

   -4.238    -7.543    -1.650    -3.936    -1.388    -3.379    -8.472     0.149


Rank of [soln1 | soln2 | soln5 | soln6 | soln7 | soln8 | soln9 | soln10] is: 5.
```

2. **(a)** The cdf of the given functions can be found out by integrating the probability distributions in the given range doing which we obtain-

2.(a) The given triangular distribution has a p.d.f. as follows

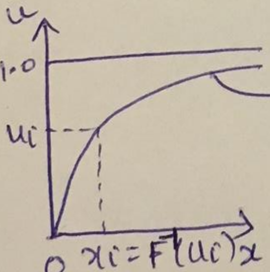
$$f(x) = \begin{cases} x & 0 \le x \le 1 \\ 2-x & 1 < x \le 2 \\ 0 & \text{otherwise} \end{cases}$$

Evaluating the cdf of $f(x)$ we get,

$$F(x) = \begin{cases} \frac{x^2}{2} & 0 \le x \le 1 \\ \frac{1}{2} + \left[2x - \frac{x^2}{2}\right]_1^x & 1 < x \le 2 \\ 0 & \text{otherwise} \end{cases}$$

Now, the inverse transform technique tells us that we can generate a sequence of u.i.i.d. random numbers $u_1, u_2, \ldots, u_n, \ldots$ in the interval $[0,1]$ [this is given], we can generate and transform these into i.i.d. samples $x_1, x_2, \ldots, x_n, \ldots$ by using inverse transform $x_i = F^{-1}(u_i)$ provided that it exists.

$\text{Prob}\{x_i \le a\} = \text{Prob}\{F^{-1}(u_i) \le a\}$
$= \text{Prob}\{u_i \le F(a)\} = F(a)$



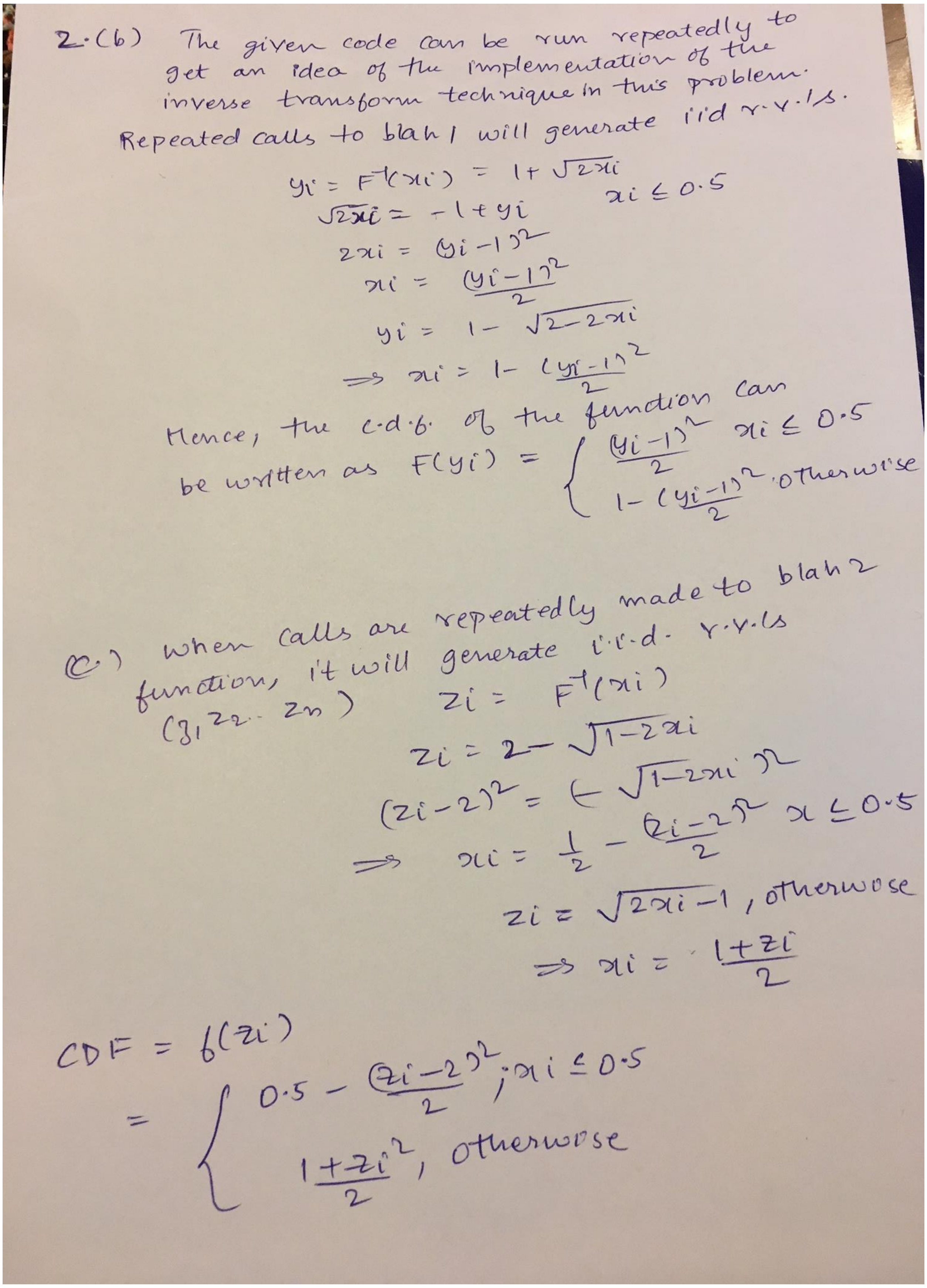
$$\Rightarrow u_i = \begin{cases} \frac{x^2}{2} & 0 \le x \le 1 \\ \frac{1}{2} + \left[2x - \frac{x^2}{2}\right]_1^x & 1 < x \le 2 \\ 0 & \text{otherwise} \end{cases}$$

$$u_i = \begin{cases} \frac{x^2}{2} & 0 \le x \le 1 \\ \frac{2-(x-2)^2}{2} & 1 \le x \le 2 \\ 0 & \text{otherwise} \end{cases}$$

Now, we need to find the values of x which would be the i.i.d's.

$$x = \begin{cases} \sqrt{2u_i} & 0 \le x \le 1 \\ \sqrt{2-2u_i} + 2 & 1 \le x \le 2 \end{cases}$$

2.(b). and 2(c).



2.(b) The given code can be run repeatedly to get an idea of the implementation of the inverse transform technique in this problem.

Repeated calls to blah1 will generate i'id r.v.'s.

$$y_i = F^{-1}(x_i) = 1 + \sqrt{2x_i} \quad x_i \le 0.5$$

$$\sqrt{2x_i} = -1 + y_i$$

$$2x_i = (y_i - 1)^2$$

$$x_i = \frac{(y_i-1)^2}{2}$$

$$y_i = 1 - \sqrt{2 - 2x_i}$$

$$\Rightarrow x_i = 1 - \frac{(y_i-1)^2}{2}$$

Hence, the c.d.f. of the function can be written as $F(y_i) = \begin{cases} \frac{(y_i-1)^2}{2} & x_i \le 0.5 \\ 1 - \frac{(y_i-1)^2}{2} & \text{otherwise} \end{cases}$

(c) when calls are repeatedly made to blah2 function, it will generate i.i.d. r.v.'s $(z_1, z_2 .. z_n)$ $z_i = F^{-1}(x_i)$

$$z_i = 2 - \sqrt{1 - 2x_i}$$

$$(z_i - 2)^2 = (-\sqrt{1-2x_i})^2$$

$$\Rightarrow x_i = \frac{1}{2} - \frac{(z_i-2)^2}{2} \quad x \le 0.5$$

$$z_i = \sqrt{2x_i - 1}, \text{ otherwise}$$

$$\Rightarrow x_i = \frac{1 + z_i}{2}$$

$$CDF = f(z_i)$$

$$= \begin{cases} 0.5 - \frac{(z_i-2)^2}{2}; & x_i \le 0.5 \\ \frac{1+z_i^2}{2}, & \text{otherwise} \end{cases}$$

3. **(a)** This is a typical example of recursion in which the outputs would be as follows-

   3a(I). kablooey is the output. As one enters the braces of the f() function, the if loop is skip until the very end when the '\n' character is actually reached(when the user actually terminates the entry of the characters by entering '\n'). While this happens, the cout keeps printing all the letters sequentially in the order they were entered and the f() is called after that and this process continues. Once '\n' is encountered, we exit the function loop. Here, it is getting a character then prints it and makes a function call.

   3a(II). yeoolbak is the output. In this case, the cout does not get executed till the time '\n' is encountered and in contrast with the part I, this time the printing out of the values starts with the last letter and keeps going backward. Here, before the character is printed the function is called once more. Hence, it gets stored in a recursive fashion before everything gets printed out. This results in the characters being printed in the reverse order when compared to the earlier case.

   3.(b). The idea of this algorithm is: if m>n, GCD (m,n) is the same as GCD(m-n,n). If m/d both leave no remainder, then (mn)/d leaves no remainder.

An example using the substitution model:

```
gcd(468, 24)
gcd(444, 24)
gcd(420, 24)
  ...
gcd(36, 24)
gcd(12, 24)      (Now n is bigger)
gcd(12, 12)      (Same)
=> 12
```

This recursive algorithm essentially takes a pair of numbers and then making a comparison between the two numbers, subtracts the smaller one from the larger one and keeps on doing this until both of them are equal or one of them reduces to 1 which eventually gives us the greatest common divisor.

Generic Proof-The validity of the Euclidean algorithm can be proven by a two-step argument. In the first step, the final nonzero remainder r_{N-1} is shown to divide both a and b. Since it is a common divisor, it must be less than or equal to the greatest common divisor g. In the second step, it is shown that any common divisor of a and b, including g, must divide r_{N-1}; therefore, g must be less than or equal to r_{N-1}. These two conclusions are inconsistent unless $r_{N-1} = g$.

To demonstrate that r_{N-1} divides both a and b (the first step), r_{N-1} divides its predecessor r_{N-2}

$$r_{N-2} = q_N r_{N-1}$$

since the final remainder r_N is zero. r_{N-1} also divides its next predecessor r_{N-3}

$$r_{N-3} = q_{N-1} r_{N-2} + r_{N-1}$$

because it divides both terms on the right-hand side of the equation. Iterating the same argument, r_{N-1} divides all the preceding remainders, including a and b. None of the preceding remainders r_{N-2}, r_{N-3}, etc. divide a and b, since they leave a remainder. Since r_{N-1} is a common divisor of a and b, $r_{N-1} \le g$.

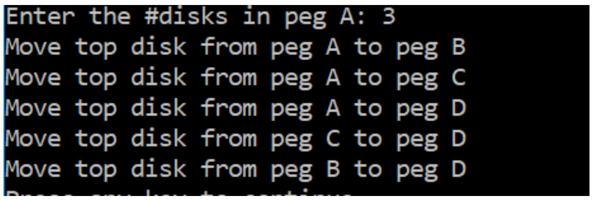
In the second step, any natural number c that divides both a and b (in other words, any common divisor of a and b) divides the remainders r_k. By definition, a and b can be written as multiples of c: $a = mc$ and $b = nc$, where m and n are natural numbers. Therefore, c divides the initial remainder r_0, since $r_0 = a - q_0b = mc - q_0nc = (m - q_0n)c$. An analogous argument shows that c also divides the subsequent remainders r_1, r_2, etc. Therefore, the greatest common divisor g must divide r_{N-1}, which implies that $g \le r_{N-1}$. Since the first part of the argument showed the reverse ($r_{N-1} \le g$), it follows that $g = r_{N-1}$. Thus, g is the greatest common divisor of all the succeeding pairs:

$$g = \gcd(a, b) = \gcd(b, r_0) = \gcd(r_0, r_1) = \ldots = \gcd(r_{N-2}, r_{N-1}) = r_{N-1}.$$

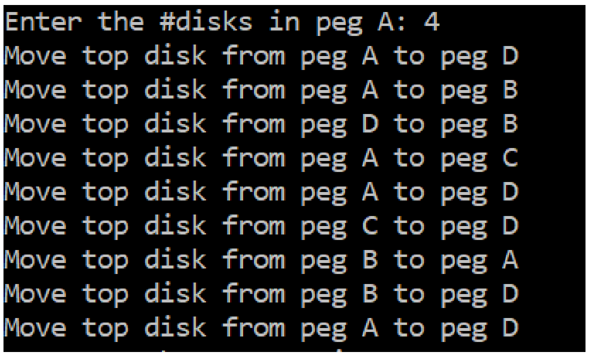
(Reference-Wikipedia Euclid's algorithm)

3.(c). EDCB The putchar() function prints the string values and in the first iteration, num%10 returns 4 which when used with putchar(4-corresponds to the 4th alphabet after A gives us E). num/10 gives us 123 which is then given as an input to putchar again and used recursively. In the second iteration, 123 is started of as an input, num%10 gives us 3 which is then input into the putchar which returns D(the 3rd letter after A) and the recursion follows through. In the next iteration, num/10 gives us 12 which when used with the function and running num%10 gives us 2 which when used with putchar gives us C(Going 2 places before A). In the final iteration, similarly we obtain B following the same line of logic and adding 1 to the final putchar call we get B, hence the output EDCB.

3.(d). There are illegal moves from n=5` to n=10.

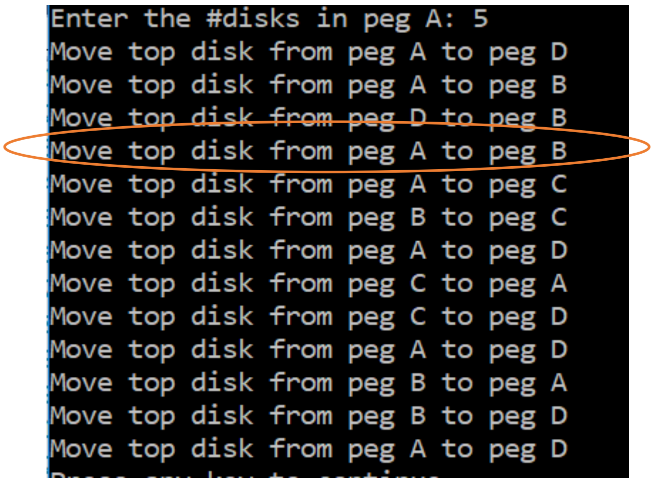


```
Enter the #disks in peg A: 3
Move top disk from peg A to peg B
Move top disk from peg A to peg C
Move top disk from peg A to peg D
Move top disk from peg C to peg D
Move top disk from peg B to peg D
```

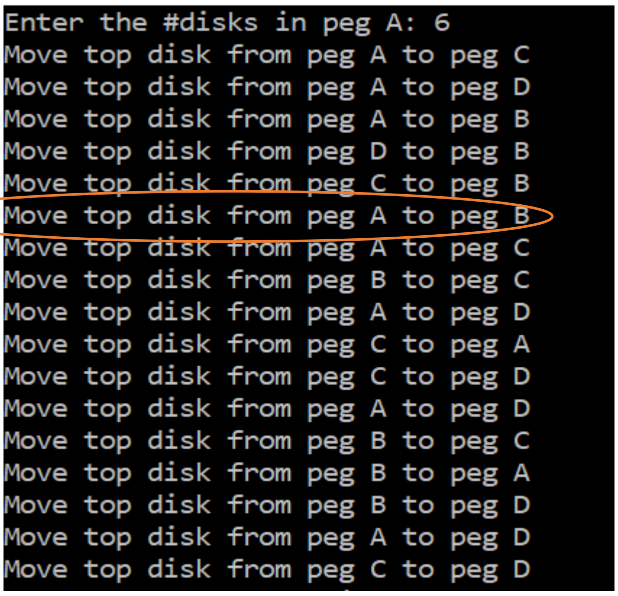


```
Enter the #disks in peg A: 4
Move top disk from peg A to peg D
Move top disk from peg A to peg B
Move top disk from peg D to peg B
Move top disk from peg A to peg C
Move top disk from peg A to peg D
Move top disk from peg C to peg D
Move top disk from peg B to peg A
Move top disk from peg B to peg D
Move top disk from peg A to peg D
```

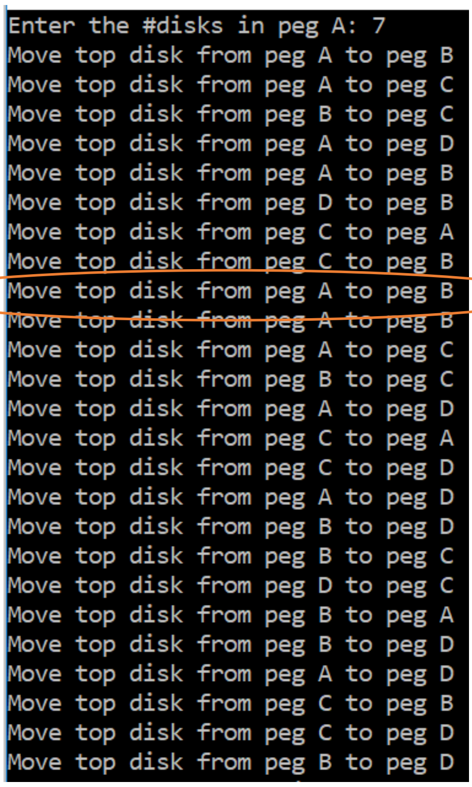
Illegal move as the move from A to B is repeated without the disk in B getting relocated after the first move. The other moves have all been circled in red for all the other cases.



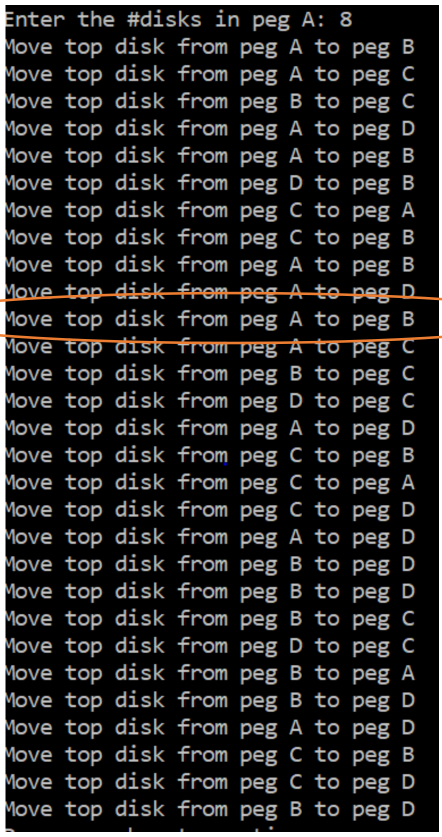
```
Enter the #disks in peg A: 5
Move top disk from peg A to peg D
Move top disk from peg A to peg B
Move top disk from peg D to peg B
Move top disk from peg A to peg B
Move top disk from peg A to peg C
Move top disk from peg B to peg C
Move top disk from peg A to peg D
Move top disk from peg C to peg A
Move top disk from peg C to peg D
Move top disk from peg A to peg D
Move top disk from peg B to peg A
Move top disk from peg B to peg D
Move top disk from peg A to peg D
```



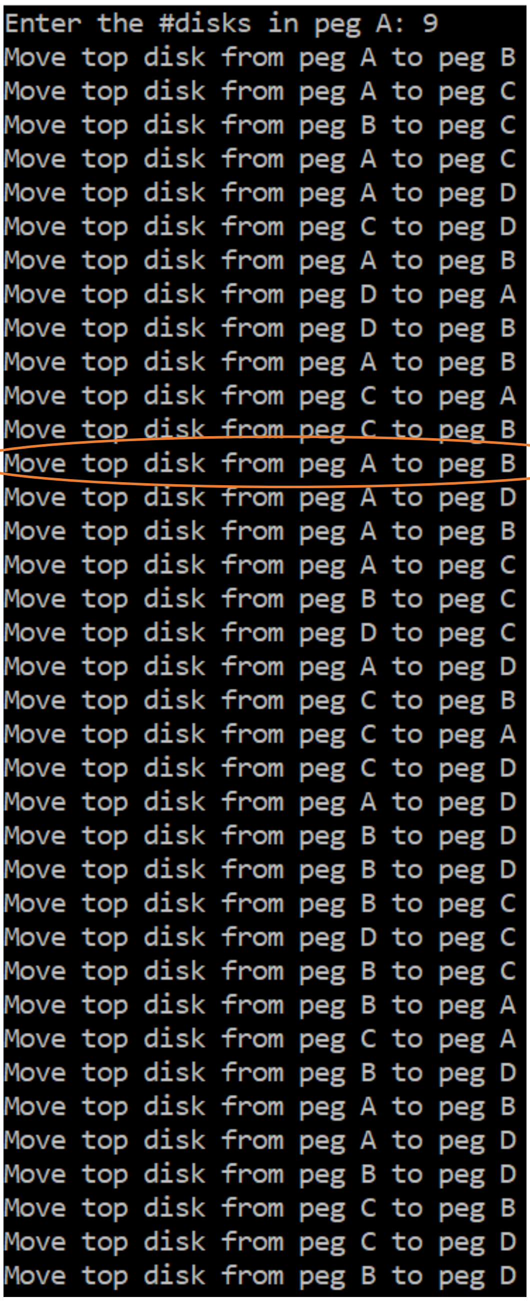
```
Enter the #disks in peg A: 6
Move top disk from peg A to peg C
Move top disk from peg A to peg D
Move top disk from peg A to peg B
Move top disk from peg D to peg B
Move top disk from peg C to peg B
Move top disk from peg A to peg B
Move top disk from peg A to peg C
Move top disk from peg B to peg C
Move top disk from peg A to peg D
Move top disk from peg C to peg A
Move top disk from peg C to peg D
Move top disk from peg A to peg D
Move top disk from peg B to peg C
Move top disk from peg B to peg A
Move top disk from peg B to peg D
Move top disk from peg A to peg D
Move top disk from peg C to peg D
```



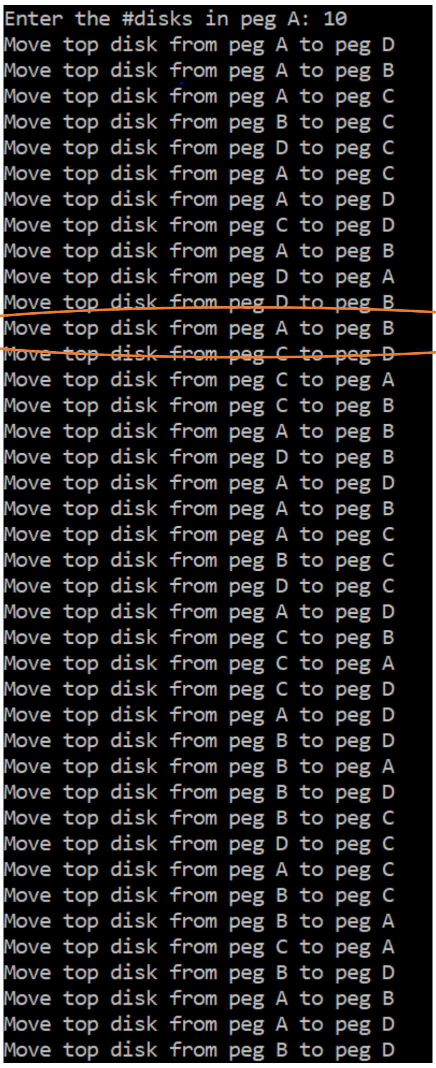
```
Enter the #disks in peg A: 7
Move top disk from peg A to peg B
Move top disk from peg A to peg C
Move top disk from peg B to peg C
Move top disk from peg A to peg D
Move top disk from peg A to peg B
Move top disk from peg D to peg B
Move top disk from peg C to peg A
Move top disk from peg C to peg B
Move top disk from peg A to peg B
Move top disk from peg A to peg B
Move top disk from peg A to peg C
Move top disk from peg B to peg C
Move top disk from peg A to peg D
Move top disk from peg C to peg A
Move top disk from peg C to peg D
Move top disk from peg A to peg D
Move top disk from peg B to peg D
Move top disk from peg B to peg C
Move top disk from peg D to peg C
Move top disk from peg B to peg A
Move top disk from peg B to peg D
Move top disk from peg A to peg D
Move top disk from peg C to peg B
Move top disk from peg C to peg D
Move top disk from peg B to peg D
```



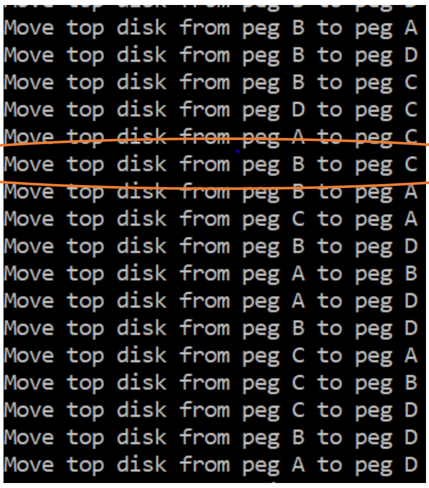
```
Enter the #disks in peg A: 8
Move top disk from peg A to peg B
Move top disk from peg A to peg C
Move top disk from peg B to peg C
Move top disk from peg A to peg D
Move top disk from peg A to peg B
Move top disk from peg D to peg B
Move top disk from peg C to peg A
Move top disk from peg C to peg B
Move top disk from peg A to peg B
Move top disk from peg A to peg D
Move top disk from peg A to peg B
Move top disk from peg A to peg C
Move top disk from peg B to peg C
Move top disk from peg D to peg C
Move top disk from peg A to peg D
Move top disk from peg C to peg B
Move top disk from peg C to peg A
Move top disk from peg C to peg D
Move top disk from peg A to peg D
Move top disk from peg B to peg D
Move top disk from peg B to peg D
Move top disk from peg B to peg C
Move top disk from peg D to peg C
Move top disk from peg B to peg A
Move top disk from peg B to peg D
Move top disk from peg A to peg D
Move top disk from peg C to peg B
Move top disk from peg C to peg D
Move top disk from peg B to peg D
```



```
Enter the #disks in peg A: 9
Move top disk from peg A to peg B
Move top disk from peg A to peg C
Move top disk from peg B to peg C
Move top disk from peg A to peg C
Move top disk from peg A to peg D
Move top disk from peg C to peg D
Move top disk from peg A to peg B
Move top disk from peg D to peg A
Move top disk from peg D to peg B
Move top disk from peg A to peg B
Move top disk from peg C to peg A
Move top disk from peg C to peg B
Move top disk from peg A to peg B
Move top disk from peg A to peg D
Move top disk from peg A to peg B
Move top disk from peg A to peg C
Move top disk from peg B to peg C
Move top disk from peg D to peg C
Move top disk from peg A to peg D
Move top disk from peg C to peg B
Move top disk from peg C to peg A
Move top disk from peg C to peg D
Move top disk from peg A to peg D
Move top disk from peg B to peg D
Move top disk from peg B to peg D
Move top disk from peg B to peg C
Move top disk from peg D to peg C
Move top disk from peg B to peg C
Move top disk from peg B to peg A
Move top disk from peg C to peg A
Move top disk from peg B to peg D
Move top disk from peg A to peg B
Move top disk from peg A to peg D
Move top disk from peg B to peg D
Move top disk from peg C to peg B
Move top disk from peg C to peg D
Move top disk from peg B to peg D
```



```
Enter the #disks in peg A: 10
Move top disk from peg A to peg D
Move top disk from peg A to peg B
Move top disk from peg A to peg C
Move top disk from peg B to peg C
Move top disk from peg D to peg C
Move top disk from peg A to peg C
Move top disk from peg A to peg D
Move top disk from peg C to peg D
Move top disk from peg A to peg B
Move top disk from peg D to peg A
Move top disk from peg D to peg B
Move top disk from peg A to peg B
Move top disk from peg C to peg D
Move top disk from peg C to peg A
Move top disk from peg C to peg B
Move top disk from peg A to peg B
Move top disk from peg D to peg B
Move top disk from peg A to peg D
Move top disk from peg A to peg B
Move top disk from peg A to peg C
Move top disk from peg B to peg C
Move top disk from peg D to peg C
Move top disk from peg A to peg D
Move top disk from peg C to peg B
Move top disk from peg C to peg A
Move top disk from peg C to peg D
Move top disk from peg A to peg D
Move top disk from peg B to peg D
Move top disk from peg B to peg A
Move top disk from peg B to peg D
Move top disk from peg B to peg C
Move top disk from peg D to peg C
Move top disk from peg A to peg C
Move top disk from peg B to peg C
Move top disk from peg B to peg A
Move top disk from peg C to peg A
Move top disk from peg B to peg D
Move top disk from peg A to peg B
Move top disk from peg A to peg D
Move top disk from peg B to peg D
```



```
Move top disk from peg B to peg A
Move top disk from peg B to peg D
Move top disk from peg B to peg C
Move top disk from peg D to peg C
Move top disk from peg A to peg C
Move top disk from peg B to peg C
Move top disk from peg B to peg A
Move top disk from peg C to peg A
Move top disk from peg B to peg D
Move top disk from peg A to peg B
Move top disk from peg A to peg D
Move top disk from peg B to peg D
Move top disk from peg C to peg A
Move top disk from peg C to peg B
Move top disk from peg C to peg D
Move top disk from peg B to peg D
Move top disk from peg A to peg D
```