1. Matrix multiplication - multiply MxN and NxL matrices using two dimensional arrays. Take any compile time constant values for M,N and L. Eg. 4x3 and 3x5. You don't need to take input from user - you can hardcode the matrix. Ouput should be the result matrix.

2. Calculate inverse of an MxN matrix.

For 3 & 4, see the following link:
https://people.richland.edu/james/lecture/m116/matrices/applications.html

3. If three points (x1,y1), (x2,y2) and (x3,y3) are given. Find,
i) if they are colinear, if so, find the length of the line
ii) if they are not colinear, if so find the area of triangle using determinant method

4. You need to write two small programs for this - one to encrypt a string and other to decrypt a string.
Encrypt.exe - It takes string as commandline parameter and outputs encrypted text.
Decrypt.exe - It takes encrypted text as commandline parameter and ouputs decrypted text.
Verify that encryption and decryption works fine.
For commandline parameters related info, see online resources (like, http://www.cplusplus.com/articles/DEN36Up4/) or take help from TAs.

5. Let's implement a horse race! Four horses (X, M, T and D) run on their tracks as depicted below (- => travelled track, . => track yet to travel and | => finish line).

Initial screen shows following state:
```
X.............................|
M.............................|
T.............................|
D.............................|
```

Choose your horse (from X, M, T and D):

After user selects the horse, you can start the race. It could progress like below.
```
-----X.............|
--------M..........|
-------T...........|
-------D...........|
```

After a horse gets past the finish line, the race stops and winner is declared. eg:
Horse M wins the race!

How do the horses run? A horse runs based on its current level of power. A rand() function increases power of horse by a random number from 1 to 5. Initial level of power for each horse is 5.
To take a step, a power increment by 5 is required. So, the horse who reaches power level of 10 or more will take the second step, 15 or more will take 3rd step and so on.

Use two dimensional matrix for keeping track of powers of horses. Feel free to use more matrix if you have to for keeping track of other states of the system. You'll write something like this (eg for two horses X & M):

```
while (theRaceIsOn) {
        power[X][stepX] = power[X][stepX-1] + rand() % 5; // add power to horse X
        power[M][stepM]= power[X][stepM-1] + rand() % 5; // add power to horse M
        if (power[X][stepX] >= power needed for next step for X)
                stepX++; // move to next step

        if (power[M][stepM] >= power needed for next step for M)
                stepM++; // move to next step

        // update display
        system ("cls");
        cout << /*display the race progress*/
}
```