1. Find and display all integers between 1 and 10000 which divide by 37. Propose at least two different ways to solve this problem.

2. Ask the user to input an integer in the range from 10 to 500. (Look up and use the `input` command.) If the input number is not an integer or is outside the limits, keep asking for a new number. Store the number in a variable *N*.

3. Write MATLAB code for the 'Guess My Number' game. First, the computer picks a random integer between 1 and 10 using the `randi` command. Next, the user is asked to enter their guess. If the guess matches the chosen number, display a congratulations message. Otherwise, display a 'better luck next time' message.

4. Consider a grid of size $n \times m$ with virtual bugs. Each bug lives in a grid cell. An example of the grid for $n = 20$ and $m = 30$ is shown in Figure 1 . The grid is given to you in a form of a matrix A of size $m \times n$, with 0s in the empty cells and 1s in the cells occupied by bugs.

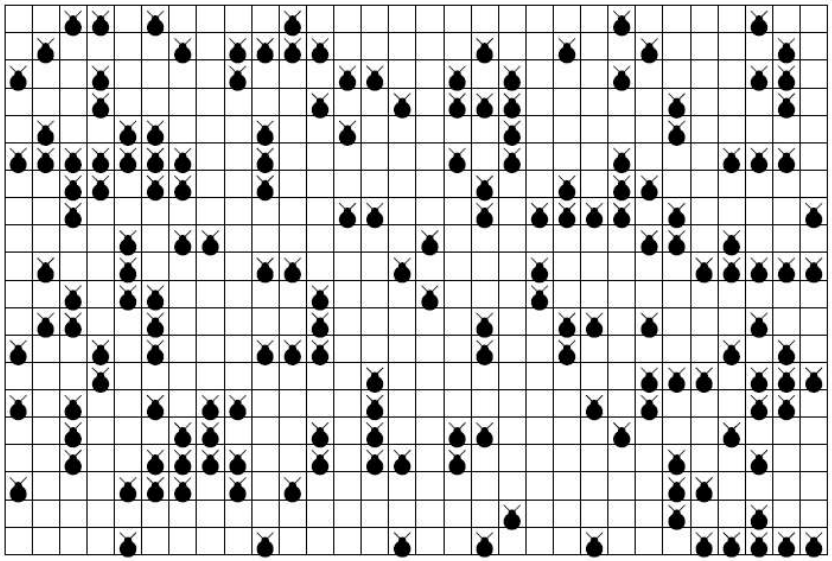

Figure 1 The virtual bugs grid.

(a) Find out the average number of neighbours per bug. Neighbours are considered to be the bugs in the 8 surrounding cells.

To demonstrate your work, create grids of different densities using:

```
A = rand(m,n) < 0.1; % sparse
A = rand(m,n) < 0.5; % medium
A = rand(m,n) < 0.7; % dense
```

(b) MATLAB includes a version of Conway's Game of Life. It is started by typing `life` in the command window. The rules are as follows:[2]

- Any bug with fewer than two neighbour bugs dies from isolation.
- Any bug with two or three neighbour bugs lives on to the next generation.
- Any bug with more than three neighbour bugs dies from overcrowding.
- Any empty cell with exactly three neighbour bugs becomes a bug, as if by reproduction.
- The rules apply in the same way to the edge and corner cells even thought they have fewer physical neighbour cells.

Using these rules, calculate the next generation of bugs for your randomly populated grid.

(c) Use the command `spy` to see the grid. Include it in a loop where you evolve the population with each pass, visualise it with `spy`, and pause the execution with command `pause(0.2)`.