# PROGRAM 9.5.1:

In [*]:
```
#AMIT CHAUHAN
#RA2311004010332 ECE-F

import random as random_number
number=random_number.randint(1,100)
attempts=0 #count no of attempts to guess the number
guess=0
while guess!=number:
    guess=eval(input('Guess a number:'))
    attempts+=1
    if guess==number:
        print('Correrct! You used',attempts,'attempts!')
        break
    elif guess<number:
        print('Go higher!')
    else:
        print('Go lower!')
```

```
Guess a number:4
Go higher!
Guess a number:67
Go higher!
Guess a number:-3
Go higher!
Guess a number:1202
Go lower!

Guess a number:
```

# PROGRAM 9.5.2:

```
5]: #AMIT CHAUHAN
    #RA2311004010332 ECE-F

    import numpy as np
    import random
    from time import sleep
    # Function to create a blank board
    def create_board():
        return np.array([[0, 0, 0],
                         [0, 0, 0],
                         [0, 0, 0]])
    # Function to get available positions on the board
    def possibilities(board):
        return [(i, j) for i in range(len(board)) for j in range(len(board)) if board[i][j] == 0]
    # Function to randomly place a player's move
    def random_place(board, player):
        selection = possibilities(board)
        current_loc = random.choice(selection)
        board[current_loc] = player
        return board
    # Function to check if a player wins in any row
    def row_win(board, player):
        for x in range(len(board)):
            if all(board[x, y] == player for y in range(len(board))):
                return True
        return False
    # Function to check if a player wins in any column
    def col_win(board, player):
        for y in range(len(board)):
            if all(board[x, y] == player for x in range(len(board))):
                return True
        return False
    # Function to check if a player wins diagonally
    def diag_win(board, player):
        if all(board[i, i] == player for i in range(len(board))):
            return True
        if all(board[i, len(board) - 1 - i] == player for i in range(len(board))):
            return True
        return False
    # Function to evaluate the game status
    def evaluate(board):
        for player in [1, 2]:
            if row_win(board, player) or col_win(board, player) or diag_win(board, player):
                return player
        if np.all(board != 0):
            return -1
        return 0
    # Main game loop
```

```
# Main game loop
def play_game():
    board = create_board()
    winner, counter = 0, 1
    print(board)
    sleep(1)
    while winner == 0:
        for player in [1, 2]:
            board = random_place(board, player)
            print(f"Board after move {counter}:")
            print(board)
            sleep(1)
            counter += 1
            winner = evaluate(board)
            if winner != 0:
                break
    return winner
# Run the game
winner = play_game()
if winner == -1:
    print("It's a draw!")
else:
    print(f"Winner is: Player {winner}")
```

```
[[0 0 0]
 [0 0 0]
 [0 0 0]]
Board after move 1:
[[1 0 0]
 [0 0 0]
 [0 0 0]]
Board after move 2:
[[1 0 2]
 [0 0 0]
 [0 0 0]]
Board after move 3:
[[1 0 2]
 [1 0 0]
 [0 0 0]]
Board after move 4:
[[1 2 2]
 [1 0 0]
 [0 0 0]]
Board after move 5:
[[1 2 2]
 [1 0 0]
 [0 0 1]]
Board after move 6:
[[1 2 2]
 [1 0 2]
 [0 0 1]]
Board after move 7:
[[1 2 2]
 [1 1 2]
 [0 0 1]]
Winner is: Player 1
```