Sara Boumrah

Problem 1

```
N= int(input("Enter a number: "))

#initializing output list
output= []

#loop values from 1 to N
for n in range(1,int(N)+1):
    output.append(n)
    repeats= N-1
    while repeats != 0: #appends further n values to output list N times
        output.append(n)
        repeats -= 1
print(output)
```

Problem 2

```
n = int(input("Enter a number n: "))  # Takes user input for n

# initialize variable for superfactorial
sf = 1

while n > 0:  # Loop until n becomes zero
    n_f = 1  # Initialize n_f for factorial calculation
    for N in range(1, n + 1):  # Computes factorial of n
        n_f *= N
    sf *= n_f  # Multiply superfactorial by calculated factorial
    n -= 1  # Decrement n in each iteration

print(sf)
```

Problem 3

```
K = 10  # Set number of wrong guesses allowed to 10

hidden_word = '______'
magic_word = 'PYTHON'

while K > 0 and hidden_word != magic_word: # Terminates when either statement is
flase
    print(hidden_word)
    guess = input("Guess a letter: ")

    found = False  # To track if the guessed letter was found in magic_word

    # Loops through magic word and checks if user guess is contained within it. If it is,
then hidden_word is edited
    for i in range(len(magic_word)):
        if guess.upper() == magic_word[i]:
            hidden_word = hidden_word[:i] + guess.upper() + hidden_word[i + 1:]
            found = True

    if not found:
        K -= 1
        print(f"Wrong guess! Remaining attempts: {K}")

if hidden_word == magic_word:
    print(f"Congratulations! You guessed the word: {hidden_word}")
else:
    print(f"Sorry, you ran out of attempts. The word was: {magic_word}")
```

Problem 4

```
# Get user input for the number
num = int(input("Enter a positive number to find its square root: "))

# Determine the initial estimate with half the number of digits
a_n = 10 ** (len(str(num)) // 2)

e = 0.00001  # Set the small value for convergence

while True:
    a_n_minus_1 = a_n
    a_n = 0.5 * (a_n_minus_1 + num / a_n_minus_1)
    if abs(a_n - a_n_minus_1) < e: # Loop continues until this statement is True
        break

print(f"The square root of {num} is approximately {a_n}")
```

Problem 5

```
data = [
    [100, 96.8, 92.6, 86.7, 78.8, 68.2, 54.4, 37.5, 21.3, 8.3],
    [96.1, 93.3, 89.2, 83.9, 76.7, 66.6, 53.5, 37.3, 21, 8.3],
    [92.2, 89.6, 85.9, 81.1, 74.2, 65, 52.7, 36.9, 21, 8.3],
    [88.2, 85.7, 82.5, 77.9, 71.7, 63.3, 51.6, 36.6, 21, 8.3],
    [84.1, 81.8, 79, 74.7, 69.1, 61.3, 50.4, 36.2, 20.8, 8.3],
    [79.9, 77.9, 75.3, 71.6, 66.4, 59.2, 49.1, 35.7, 20.8, 8.3],
    [75.4, 73.7, 71.4, 68, 63.4, 56.9, 47.7, 35.2, 20.8, 8.3],
    [71, 69.4, 67.3, 64.5, 60.4, 54.4, 46.1, 34.5, 20.7, 8.3],
    [66.4, 65, 63.3, 60.6, 57.1, 51.9, 44.3, 33.6, 20.5, 8.3],
    [61.7, 60.4, 59, 56.7, 53.7, 49.1, 42.4, 32.7, 20.3, 8.3],
    [56.7, 55.8, 54.4, 52.7, 50, 46.1, 40.3, 31.6, 20.1, 8.3],
    [51.8, 51.1, 49.8, 48.4, 46.1, 42.8, 37.8, 30.2, 19.8, 8.3],
    [46.6, 45.9, 45.1, 43.8, 42, 39.4, 35.2, 28.6, 19.3, 8.3],
    [41.3, 40.8, 40.1, 39.2, 37.8, 35.5, 32.2, 26.9, 18.6, 8.3],
    [35.9, 35.5, 35, 34.3, 33.2, 31.4, 29, 24.6, 17.8, 8.1],
    [30.4, 30, 29.7, 29.2, 28.4, 27.2, 25.3, 22.1, 16.6, 8.1],
    [24.6, 24.4, 24.2, 23.9, 23.3, 22.4, 21.2, 18.9, 14.8, 8],
    [18.7, 18.6, 18.4, 18.2, 18, 17.5, 16.8, 15.4, 12.7, 7.4],
    [12.7, 12.5, 12.5, 12.4, 12.4, 12, 11.7, 11, 9.7, 6.5],
    [6.4, 6.4, 6.4, 6.4, 6.4, 6.2, 6.2, 6, 5.7, 4.4]
]

# Initialize user input variables
overs_t1_played= int(input("Enter the number of overs played by Team 1: "))
totalrun_t1= int(input("Enter Team 1's total run: "))
wicketloss_t1= int(input("Enter Team 1's wicket loss: "))
overs_t2_available= int(input("Enter the number of overs available for Team 2: "))

# Calculate resources for Team 1 and 2
resource_team1 = data[20 - overs_t1_played][wicketloss_t1] # Accessing resource data points from matrix
resource_team2 = data[20 - overs_t2_available][0]  # Team 2 always starts with 0 wickets lost

# Calculate revised target for Team 2
revised_target = (totalrun_t1 * (resource_team2 / resource_team1))

print(f"Revised target for Team 2 in {overs_t2_available} overs: {revised_target}")
```

Problem 6

```
urls = [
    "https://zslskwqbi.com",
    "https://nuflpogyxb.com",
    "https://xxbtkaewit.com",
    "https://wneeroj.com",
    "https://uiomtwwqu.com",
    "https://irexzwynlr.com",
    "https://gykevxk.com",
    "https://pxhjwpt.com",
    "https://bmedahrwm.com",
    "https://djgwuqi.com",
    "https://uqycuitvq.com",
    "https://munucbyxgt.com",
    "https://sliho.com",
    "https://bxmkcd.com",
    "https://uqddpxglf.com",
    "https://muuuwlyyi.com",
    "https://ymshlq.com",
    "https://bhbyyjqx.com",
    "https://gotesq.com",
    "https://ciqqg.com",
    "https://skkpl.com",
    "https://uwrapqf.com",
    "https://kexcka.com",
    "https://scouyemwgo.com",
    "https://twoix.com",
    "https://lfebzakott.com",
    "https://fcpebs.com",
    "https://gotesq.com",
    "https://munucbyxgt.com",
    "https://scouyemwgo.com",
    "https://skkpl.com",
    "https://nuflpogyxb.com",
    "https://fcpebs.com",
    "https://munucbyxgt.com",
    "https://khnrmumxeu.com",
    "https://nuflpogyxb.com",
    "https://fcpebs.com",
    "https://munucbyxgt.com",
    "https://khnrmumxeu.com",
    "https://nuflpogyxb.com",
    "https://pxhjwpt.com",
    "https://bmedahrwm.com",
    "https://djgwuqi.com",
    "https://uqycuitvq.com",
    "https://lfebzakott.com",
    "https://fcpebs.com",
    "https://gotesq.com",
    "https://munucbyxgt.com",
    "https://skkpl.com",
    "https://nuflpogyxb.com",
    # ^Randomly generated URLs
]


# Count occurrences of each URL
url_counts = {}
for url in urls:
    if url in url_counts:
        url_counts[url] += 1
    else:
        url_counts[url] = 1


top_five_urls = [] # Initialize an empty list to store the top five URLs

# Iterate five times to find the top five URLs
for i in range(5):
    # Initialize values at the start of each iteration
    max_count = 0
    max_url = ""
    # Loop through each URL count in the dictionary
    for url, count in url_counts.items():
        if count > max_count and url not in top_five_urls:  # Check if the count is higher
than the current maximum
            max_count = count  # Update the maximum count if a higher count is found
            max_url = url  # Update the URL corresponding to the new maximum count
    if max_url:
        top_five_urls.append(max_url)  # Add the URL with the highest count to the
top_five_urls list

# Show the top five URLs
print("Top five visited URLs:")
count = 1  # Initialize a counter to display the top URLs
while count <= 5:  # Loop through the top five URLs
    for url in top_five_urls:
        print(f"{count}. {url}")
        count += 1
        if count > 5:  # Break the loop if the counter exceeds 5
            break
```

Problem 7

```
# Create an empty 3x3 game board
board = [['_','_','_'],
         ['_','_','_'],
         ['_','_','_']
        ]

# Display the initial empty board
for i in range(3):
    for j in range(3):
        print(board[i][j], end = " ")
    print()
print()

# Game logic
current_player = 'X'  # Initialize player X as the starting player
GameOver = False  # Initialize the game state as not over

# Run the game loop until the game is over
while not GameOver:
    print(f"Player {current_player}'s turn.")
    row = int(input("Enter row (1, 2, or 3): "))  # Get user input for row
    col = int(input("Enter column (1, 2, or 3): "))  # Get user input for column

    # Check if the selected spot on the board is empty
    if board[row-1][col-1] == '_':
        board[row-1][col-1] = current_player  # Place the player's symbol on the board

        # Display the updated board
        for i in range(3):
            for j in range(3):
                print(board[i][j], end = " ")
            print()
        print()

        # Check for a win in rows, columns, or diagonals
        for i in range(3):
            if board[i][0] == board[i][1] == board[i][2] == current_player or board[0][i] ==
board[1][i] == board[2][i] == current_player:
                print(f"Player {current_player} wins!")
                GameOver = True
                break
        if board[0][0] == board[1][1] == board[2][2] == current_player or board[0][2] ==
board[1][1] == board[2][0] == current_player:
            print(f"Player {current_player} wins!")
            GameOver = True
            break

        # Check for a tie
        is_tie = True
        for row in board:
            for cell in row:
                if cell == '_':
                    is_tie = False
                    break
            if not is_tie:
                break

        # If it's a tie, end the game
        if is_tie:
            print("It's a tie! Game over.")
            GameOver = True
            break

        # Switch players for the next turn
        if current_player == 'X':
            current_player = 'O'
        else:
            current_player = 'X'
    else:
        print("That spot is already taken. Try again.")  # If the spot is already filled, prompt
for another move


print("Congratulations! You have completed the game ")
```