APP - WEEK 11

Name - Adya Singh
RA2211003010181

```
1) def find_largest(arr):
    largest = arr[0]
    second_largest = arr[1]
    for i in range(2, len(arr)):
        if arr[i] > largest:
            second_largest = largest
            largest = arr[i]
        elif arr[i] > second_largest:
            second_largest = arr[i]
    return largest, second_largest

# Example usage
arr = [10, 20, 4, 45, 99]
largest, second_largest = find_largest(arr)
print("First largest number:", largest)
print("Second largest number:", second_largest)
```

```
2) def sum_even_odd(arr):
    even_sum = 0
    odd_sum = 0
    for num in arr:
        if num % 2 == 0:
            even_sum += num
        else:
            odd_sum += num
    return even_sum, odd_sum

# Example usage
arr = [1, 2, 3, 4, 5, 6, 7, 8, 9]
even_sum, odd_sum = sum_even_odd(arr)
print("Sum of even numbers:", even_sum)
print("Sum of odd numbers:", odd_sum)
```

```
3) def count_occurrences(arr, num):
    count = 0
    for i in arr:
        if i == num:
            count += 1
    return count

# Example usage
arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 10, 10]
num = 10
occurrences = count_occurrences(arr, num)
print("Number of occurrences of", num, "in the array:", occurrences)
```

```
4) def is_palindrome(word):
    return word == word[::-1]

def find_palindromes(sentence):
    palindromes = []
    words = sentence.split()
    for word in words:
        if is_palindrome(word):
            palindromes.append(word)
    return palindromes

# Example usage
sentence = "A man a plan a canal Panama"
palindromes = find_palindromes(sentence)
print("Palindromic words in the sentence:", palindromes)
```

```
5) def remove_duplicates(arr):
    seen = set()
    result = []
    for num in arr:
        if num not in seen:
            seen.add(num)
            result.append(num)
    return result

# Example usage
arr = [1, 2, 3, 2, 1, 4, 5, 4, 6, 7, 6, 8, 9]
unique_arr = remove_duplicates(arr)
print("List with duplicates removed:", unique_arr)
```

```
6) def matrix_multiplication(A, B):
    if len(A[0]) != len(B):
        print("Matrices are not compatible for multiplication")
        return None
    C = [[0 for _ in range(len(B[0]))] for _ in range(len(A))]
    for i in range(len(A)):
        for j in range(len(B[0])):
            for k in range(len(B)):
                C[i][j] += A[i][k] * B[k][j]
    return C

# Example usage
A = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
B = [[9, 8, 7], [6, 5, 4], [3, 2, 1]]
C = matrix_multiplication(A, B)
if C is not None:
    print("Result of matrix multiplication:")
    for row in C:
        print(row)
```

```
7) n = 3  # Change this value to adjust the size of the diamond

# Upper half of the diamond
for i in range(1, n + 1):
    for j in range(1, i * 2):
        print(j, end="")
    print()

# Lower half of the diamond
for i in range(n - 1, 0, -1):
    for j in range(1, i * 2):
        print(j, end="")
    print()
```

```
8) import random

# Generate a random number between 1 and 100
random_number = random.randint(1, 100)

attempts = 0

while True:
    try:
        user_guess = int(input("Guess the number (between 1 and 100): "))
        attempts += 1
        if user_guess < random_number:
            print("Too low! Try again.")
        elif user_guess > random_number:
            print("Too high! Try again.")
        else:
            print(f"Congratulations! You guessed the number {random_number} correctly in
{attempts} attempts.")
            break
    except ValueError:
        print("Please enter a valid number between 1 and 100.")
```

```
9) import re

def is_strong_password(password):
    # Define evaluation criteria
    length_criteria = len(password) >= 8
    lowercase_criteria = any(char.islower() for char in password)
    uppercase_criteria = any(char.isupper() for char in password)
    digit_criteria = any(char.isdigit() for char in password)
    special_character_criteria = re.search(r'[!@#$%^&*()_+{}\[\]:;<>,.?~\\-]', password) is not
None

    criteria = [length_criteria, lowercase_criteria, uppercase_criteria, digit_criteria,
special_character_criteria]

    return all(criteria)

# Get a password from the user
password = input("Enter a password: ")

# Check the strength of the password
if is_strong_password(password):
    print("Password is strong. Good job!")
else:
    print("Password is not strong. Please consider the following:")

    if not length_criteria:
        print("- Password should be at least 8 characters long")
    if not lowercase_criteria:
        print("- Password should contain at least one lowercase letter")
    if not uppercase_criteria:
        print("- Password should contain at least one uppercase letter")
```

```
  if not digit_criteria:
    print("- Password should contain at least one digit")
  if not special_character_criteria:
    print("- Password should contain at least one special character
(!@#$%^&*()_+{}[]:;<>,.?~\\-)")
```

```
10) def generate_fibonacci(n):
  if n <= 0:
    return []
  elif n == 1:
    return [0]
  elif n == 2:
    return [0, 1]
  else:
    fibonacci_sequence = [0, 1]
    while len(fibonacci_sequence) < n:
      next_number = fibonacci_sequence[-1] + fibonacci_sequence[-2]
      fibonacci_sequence.append(next_number)
    return fibonacci_sequence

# Get the number of terms from the user
n_terms = int(input("Enter the number of Fibonacci terms to generate: "))

if n_terms <= 0:
  print("Please enter a positive number of terms.")
else:
  fibonacci_sequence = generate_fibonacci(n_terms)
  print(f"Fibonacci sequence with {n_terms} terms: {fibonacci_sequence}")
```