## **Assignment Questions**

**Question 1:** Write a Python function that checks if a given string is a palindrome. A palindrome is a word, phrase, number, or other sequence of characters that reads the same forward and backward. The function should return `True` if the string is a palindrome and `False` otherwise.

Input: ‘abdba'  
Expected Output: True

**Question 2:** Implement a function that computes the factorial of a given integer using recursion. The function should take an integer as input and return its factorial.

Input: 8  
Expected Output: 40320

**Question 3:** Write a Python program to check Armstrong’s number. The armstrong number can be defined as n-digit numbers equal to the sum of the nth powers of their digits are called armstrong numbers. Print whether the given number is armstrong number or not

Armstrong Numbers: 153, 370, 371, 407, 1634, 8208, 9474

Input: 371  
Expected Output: True

**Question 4:** Write a Python program to generate the Fibonacci sequence up to a specified number of terms. The program should allow the user to input the number of terms and then print the Fibonacci sequence.

Input: 5  
Expected Output: [0, 1, 1, 2, 3]

**Question 5:** Create a function that takes a list of integers as input and returns the maximum sum of any contiguous subarray (a subarray is a contiguous portion of the array). The function should have a time complexity of O(n).

Input: [1, 2, 4, 9, 8]  
Expected Output: 24

**Question 6:** Implement a function to find the first non-repeating character in a given string. If all characters repeat, return None.

Input: ‘bosscoder academy’  
Expected Output: ‘b’

**Question 7:** Given a list of integers, write a Python function to find the maximum sum of a subarray with the constraint that no two elements in the subarray are adjacent.

Input: [1, 3, 4, 9, 8]  
Expected Output: 13

**Question 8:** Create a function that takes two lists as input and returns a new list containing the common elements of the two input lists.

Input:   
list1=[6, 7, 8, 0, -9, 1]  
list2=[1, 6, 0, 3, 4, 5, 7, 6]  
Expected Output:  
[0, 1, 6, 7]

**Question 9:** Implement a function that checks if a given string has balanced parentheses. The function should return `True` if the parentheses are balanced, and `False` otherwise. For example, "([]){}" is balanced, but "([)]" is not.

Input: ‘{a + (b - c)] \* (x / y)}’  
Expected Output: False

**Question 10:** Write a Python program to implement a basic calculator that can perform addition, subtraction, multiplication, and division. The program should take input in the form of a string, e.g., "2 + 3", and return the result of the operation.

Input: '3+2\*5'  
Output: 13

## 

## **Assignment Solutions**

**Solution 1**

def is\_palindrome(s):

s = s.lower().replace(" ", "")

return s == s[::-1]

**Solution 2:**

def factorial(n):

if n == 0:

return 1

else:

return n\*factorial(n - 1)

**Solution 3:**

num=int(input("Enter the number to check armstrong number: "))

arms = num

length = len(str(num))

sum1 = 0

while num != 0:

rem = num % 10

sum1 = sum1+(rem\*\*length)

num = num//10

if arms == sum1:

print("The given number", arms, "is armstrong number")

else:

print("The given number", arms, "is not an armstrong number")

**Solution 4:**

def fibonacci\_sequence(n):

a, b = 0, 1

fib\_sequence = []

while len(fib\_sequence) < n:

fib\_sequence.append(a)

a, b = b, a + b

return fib\_sequence

**Solution 5:**

def max\_subarray\_sum(nums):

max\_sum = current\_sum = nums[0]

for num in nums[1:]:

current\_sum = max(num, current\_sum + num)

max\_sum = max(max\_sum, current\_sum)

return max\_sum

**Solution 6:**

def first\_non\_repeating\_char(s):

char\_count = {}

for char in s:

if char in char\_count:

char\_count[char] += 1

else:

char\_count[char] = 1

for char in s:

if char\_count[char] == 1:

return char

return None

**Solution 7:**

def max\_sum\_non\_adjacent(nums):

incl = 0

excl = 0

for num in nums:

new\_excl = max(incl, excl)

incl = excl + num

excl = new\_excl

return max(incl, excl)

**Solution 8:**

def common\_elements(list1, list2):

return list(set(list1) & set(list2))

**Solution 9:**

def balanced\_parentheses(s):

stack = []

open\_brackets = "([{"

close\_brackets = ")]}"

bracket\_pairs = {")": "(", "}": "{", "]": "["}

for char in s:

if char in open\_brackets:

stack.append(char)

elif char in close\_brackets:

if not stack or stack.pop() != bracket\_pairs[char]:

return False

return not stack

**Solution 10:**

def basic\_calculator(expr):

try:

result = eval(expr)

return result

except:

return "Invalid input"