QUESTION 1

# Create two separate lists for Catholic and Anglican martyrs.

catholic\_martyrs = [

    "Achileo Kiwanuka", "Adolphus Ludigo-Mukasa", "Ambrosius Kibuuka", "Anatoli Kiriggwajjo",

    "Andrew Kaggwa", "Antanansio Bazzekuketta", "Bruno Sserunkuuma", "Charles Lwanga",

    "Denis Ssebuggwawo Wasswa", "Gonzaga Gonza", "Gyavira Musoke", "Yowana Mukiibi",

    "Yusufu Lugalama", "Zakayo Lubega", "James Buuzaabalyaawo", "John Maria Muzeeyi",

    "Joseph Mukasa Kizito", "Lukka Baanabakintu", "Matiya Mulumba", "Mbaga Tuzinde",

    "Mugagga Lubowa", "Mukasa Kiriwawanvu", "Nowa Mawaggali", "Ponsiano Ngondwe"

]

anglican\_martyrs = [

    "Aaron Lubega", "Apollo Kivebulaya", "Eria Sebukyala", "Fredrick Kizza", "George Kizza",

    "James Hannington", "Janani Luwum", "Joseph Balikuddembe", "Kizito",

    "Mark Kakumba", "Matia Mulumba", "Nuhu Mbegu", "Robert Munyagayirwa", "Samwiri Mukasa",

    "Yefusa Namayanja", "Yohana Mukasa", "Yosefu Lugalama", "Yowana Kitaka", "Yowana Maria Mukasa"

]

# Remove duplicate names that are present in both lists.

catholic\_martyrs = list(set(catholic\_martyrs) - set(anglican\_martyrs))

# Define a function to determine the group of the martyr.

def martyr\_count(name):

    if name in catholic\_martyrs:

        return "Catholic"

    elif name in anglican\_martyrs:

        return "Anglican"

    else:

        return "Not Found"

# Determine the group of the martyr named "Kizito".

kizito\_group = martyr\_count("Kizito")

# Function to check if a given string matches any of the martyr names in both lists.

def is\_martyr\_name(name):

    return name in catholic\_martyrs or name in anglican\_martyrs

# Example usage of the function:

input\_name = "Mugagga Lubowa"

result = is\_martyr\_name(input\_name)

print(result) # Create two separate lists for Catholic and Anglican martyrs.

catholic\_martyrs = [

    "Achileo Kiwanuka", "Adolphus Ludigo-Mukasa", "Ambrosius Kibuuka", "Anatoli Kiriggwajjo",

    "Andrew Kaggwa", "Antanansio Bazzekuketta", "Bruno Sserunkuuma", "Charles Lwanga",

    "Denis Ssebuggwawo Wasswa", "Gonzaga Gonza", "Gyavira Musoke", "Yowana Mukiibi",

    "Yusufu Lugalama", "Zakayo Lubega", "James Buuzaabalyaawo", "John Maria Muzeeyi",

    "Joseph Mukasa Kizito", "Lukka Baanabakintu", "Matiya Mulumba", "Mbaga Tuzinde",

    "Mugagga Lubowa", "Mukasa Kiriwawanvu", "Nowa Mawaggali", "Ponsiano Ngondwe"

]

anglican\_martyrs = [

    "Aaron Lubega", "Apollo Kivebulaya", "Eria Sebukyala", "Fredrick Kizza", "George Kizza",

    "James Hannington", "Janani Luwum", "Joseph Balikuddembe", "Kizito",

    "Mark Kakumba", "Matia Mulumba", "Nuhu Mbegu", "Robert Munyagayirwa", "Samwiri Mukasa",

    "Yefusa Namayanja", "Yohana Mukasa", "Yosefu Lugalama", "Yowana Kitaka", "Yowana Maria Mukasa"

]

# Remove duplicate names that are present in both lists.

catholic\_martyrs = list(set(catholic\_martyrs) - set(anglican\_martyrs))

# Define a function to determine the group of the martyr.

def martyr\_count(name):

    if name in catholic\_martyrs:

        return "Catholic"

    elif name in anglican\_martyrs:

        return "Anglican"

    else:

        return "Not Found"

# Determine the group of the martyr named "Kizito".

kizito\_group = martyr\_count("Kizito")

# Function to check if a given string matches any of the martyr names in both lists.

def is\_martyr\_name(name):

    return name in catholic\_martyrs or name in anglican\_martyrs

# Example usage of the function:

input\_name = "Mugagga Lubowa"

result = is\_martyr\_name(input\_name)

print(result)

QUESTION 2

a) The data structure being used in this code is a list. Specifically, it's a one-dimensional list of integers. This data structure is suitable for the task because it allows for easy storage and manipulation of a sequence of numbers.

b) initializing **maxSum** provides a starting point for comparison. This ensures that even if all the elements in the list are negative, the algorithm will still return the maximum subarray sum.

c) The for loop iterates through the list from the first element to the last. The variable **sumz** is used to keep track of the running sum of subarrays. It is initially set to 0 and then updated by adding the current element in each iterates.

d) The code uses Kadane's algorithm to identify the maximum subarray sum. The **sumz** variable keeps track of the sum of the current subarray. If **sumz** becomes negative, it means that the current subarray should be discarded, so **sumz** is reset to 0. If **sumz** becomes greater than **maxSum**, it means that a new maximum subarray has been found, so **maxSum** is updated.

e) The time complexity of this code is O(n), where n is the length of the input list. This is because the code uses a single pass through the list, performing constant-time operations (such as comparisons and additions) for each element.

The choice of using a list to store the numbers is appropriate for this task, as it allows easy access to individual elements. The algorithm (Kadane's algorithm) itself is designed to efficiently find the maximum subarray sum in a single pass through the list, which contributes to the overall efficiency of the code.

Objectives;

1.c

2.b

3.b

4.a,b

5.c

6.c

7.a

8.b

9.a

10.a