ASINGWIIRE ENOCH
M23BB38/027

Part a The data structure being used to store numbers in the code is a list. Thos data structure is suitable for this purpose because this allows a user to store mutiple values in order which therefore allows for easy access by use of indices and manipulation of this data stored in the list.

Part b Initialising the maxsum with the first element of the list ensures that the algorithm will find at least one subarray with a positive sum implying that if the first element of the list is negative, then the algorithm will not find any subarrays with a positive sum if the maxsum was not initialis

Part c The 'for -loop' iterates through the list in order starting from the beginning of the list to the end of the list . the sumz variable is used to store the running sum of the elements in the list as the loop repeats itself. The variable is then updated within the loop by adding the current number in the list to the sumz variable.

Part d The code identifies the maximum subarray sum by comparing the current sum I.e the sumz to the maximum sum I.e the maxsum. If the current sum is greater than the maximum sum, then the maximum sum is updated to the current s

Part e The time complexity of the code for finding the maximum subarray is 0(n) where n is the number of elements in the list (size) . the choice of data structure and algorithm that is to say using a list and a linear iteration through the elements contributes to the efficiency of the program. this is so due to the fact that the algorithm does not need complex data structures or nested iterations hence resulting in a linear time complexity making it easier and more efficient

C.append()
1. 2. B .my_list.insert(0,'x')
2. 3. B. [111,7,2,1,4]
3. 4. D. All the above
4. 5. C. my_list.pop(0)
5. 6. C. makes|1 and |2 point to the same list
6. 7. A. 'apple' not in fruits
7. 8. B . (3,4,5)
8. 9. A. My_list[-2]
9. 10. .A .true