|  |  |
| --- | --- |
| 1. Look for the index of the given element x in the given array:  X = [22,2,1,7,11,13,5,2,9]    SearchA(Arr, x) – return array of indices    Arr: Array  x: element to be searched | **Input**: Enter the number: 2  **Output**: Index: 1,7 |
| **Solution:**  def SearchA(Arr, x):  ind = []  for i in range(len(Arr)): # Searching element wise  if Arr[i] == x: # Found  ind.append(i)  if not ind:  return 'Element not found'  else:  return ind  arr = [22,2,1,7,11,13,5,2,9]  x = 2  print (SearchA(Arr, x)) | |
| 2. Answer question 1 in the scenario where the input array is already sorted.  How much elements you need to check in sorted array.    SearchB(Arr, x)-- return array of indices    Arr: Array  x: element to be searched | **Input**: Enter the number: 2  **Output**: Index: 1,7 |
| **Solution:**  In worst case it is . Means if you have 8 elements in array, you need to guess max 4 elements. | |
| 3. Write a function that takes an array as input, starting and ending index and return the index of minimum element from start to ending index in the array.    Minimum(Arr, starting, ending)— return integer | For example, you are given the following inputs  Array: [3,4,7,8,0,1,23,-2,-5]  StartingIndex: 4  EndingIndex: 7    Output: (Return index of minimum element) 7 |
| **Solution:**  def Minimum (arr, start, end):  min = arr[start]  ind = start  for i in range (start+1, end+1): # Searching Element wise  if arr[i] < min:  min = arr[i]  ind = i  return ind  arr = [3,4,7,8,0,1,23, -2, -5]  start = 4  end = 7  print (Minimum (arr, start, end)) | |
| **4.** Sort an array X using the above generated function.    **Hint:** Find the smallest element from the unsorted part of the array repeatedly and place it at the start of the array.    Sort4(Arr)—return array  Arr: Array to be sorted | **Output**: X = [-5, -4, -3, 0, 1, 1, 4, 35, 100, 101] |
| **Solution:**  def Sort4 (arr):  for i in range(len(arr)):  min = i  for j in range (i+1, len(arr)):  if arr[j] < arr[min]:  min = j  arr[i], arr[min] = arr[min], arr[i]  return arr  arr = [-5, -4, -3, 0, 1, 1, 4, 35, 100, 101]  print(Sort4 (arr)) | |
| 5. Extract the relevant portion and print it in the reverse direction from the string s = **"University of Engineering and Technology Lahore"**.  Without using any loop and reverse () method.      StringReverse(str, starting, ending)—returns string | **Output**: "ygolonhceT dn" |
| **Solution:**  def StringReverse (str, starting, ending):  if (starting >= ending or starting < 0 or ending > len(str)):  return 'Indexes has no sense'  if starting == 0:  starting = None  if ending == 0:  ending = None  if starting == None or ending == None:  return str [ending: starting:-1]  return str [ending-1: starting-1: -1]  s = "University of Engineering and Technology Lahore"  print (StringReverse (s, 27, 40)) | |
| 6. Given a number, the task is to find the sum of its digits using an iterative and recursive method.    SumIterative(number) – returns integer    SumRecursive(number)-- returns integer | **Input**: 1524  **Output**: Sum of digits is: 12 |
| **Solution:**   1. **Iterative Sum**   def SumIterative (num):  sum = 0  num = str(num)  while len(num) > 0:  sum += int (num [-1])  num = num [: len(num) - 1]  return sum  num = 112  print(sumIteratively(num))   1. **Recursive Sum**   def SumRecursive (num):  if num == 0:  return 0  return num % 10 + SumRecursive (num // 10)  print (SumRecursive (112)) | |
| 7. Find the sum of the given matrix both column- and row-wise.  A =  ColumnWiseSum(Mat) – returns 1d array  RowWiseSum(Mat) – returns 1d array | 27  **Output**: Row-wise: 22  17      Column-wise: 10 28 28 |
| **Solution:**   1. **Row wise sum**   def RowWiseSum (mat):  result = [0] \* len(mat)  for i in range(len(mat)):  sum = 0  for j in range(len(mat[i])):  sum += mat[i][j]  result[i] = sum  return result  mat = [[11, 12, 5, 2], [15, 6, 10], [10, 8, 12, 5]]  print (RowWiseSum (mat))   1. **Col wise sum**   def ColumnWiseSum (mat):  result = [0] \* len (mat [0])  for i in range (len (mat [0])):  sum = 0  for j in range(len(mat)):  sum += mat[j][i]  result[i] = sum  return result  mat = [[11, 12, 5, 2], [15, 6, 2, 10], [10, 8, 12, 5]]  print (ColumnWiseSum(mat)) | |
| 8. Without using any sorting methods, combine two sorted arrays keeping the resultant array sorted in ascending order.     1. = [0,3,4,10,11] 2. = [1,8,13,24]     SortedMerge(Arr1, Arr2) – returns sorted array | **Output:** [0,1,3,4,8,10,11,13,24] |
| **Solution:**  def SortedMerge (arr1, arr2):  arr = []  while len(arr1) > 0 and len(arr2) > 0:  if arr1[0] < arr2[0]:  arr. append (arr1[0])  arr1.pop(0)  else:  arr.append(arr2[0])  arr2.pop(0)  if len(arr1) > 0:  [arr.append(arr1[i]) for i in range(len(arr1))]  if len(arr2) > 0:  [arr.append(arr2[i]) for i in range(len(arr2))]  return arr  arr1 = [1, 3, 5, 7, 9, 13, 14, 15, 16, 17, 18, 19]  arr2 = [2, 4, 6, 8, 10]  print(SortedMerge (arr1, arr2)) | |
| 9. Write a recursive function that takes a string  and returns if the string is palindrome or not.    PalindromRecursive(str)- returns a boolean | **Input:** "radar"  **Output:** Palindrome |
| **Solution:**  def PalindromRecursive (str):  if len(str) == 0:  return True  if str [0] != str[len(str)-1]:  return False  return PalindromRecursive (str[1:-1])  str = 'RADAR'  print(PalindromRecursive (str)) | |
| 10. Sort the given array so that the elements are arranged in the following way while taking ascending order into consideration  Sort10(Arr)—returns array | **Input:** [10, -1, 9, 20, -3, -8, 22, 9, 7]  **Output:** [-8, 7, -3, 9, -1, 9, 10, 20, 22] |
| **Solution:**  def Sort10 (arr):  result = []  i = 0  while arr:  if i%2 == 0:  element = min([arr[x] for x in range( len(arr)) if arr[x] < 0 ] , default=min(arr))  arr.remove(element)  result.append(element)  else:  element = min([arr[x] for x in range( len(arr)) if arr[x] >= 0 ], default=min(arr))  arr.remove(element)  result.append(element)  i += 1  return result  arr = [-22, -8, 20, -9, -1, 9, -1, -9, -3, 7, 10]  print(Sort10 (arr)) | |