**DSA Lab One(Array)**

From the code written, the assignment was to create three functions, find\_max number, count the number of times a number repeats and check if an array is sorted or not.

**For function 1(find\_max element in an array):**

This function is used to check the largest number in an array

In the code below an if statement is used, its used to check if the array(arr) is indeed an array or not. If it’s not then the code stops running and a prompt saying “An array with numbers is needed”. But if the a proper array is given then the code continues.

def find\_max(arr):

    if not arr:

        return "An array with numbers is needed"

    else:

        max\_ele = arr[0]

        for element in arr:

            if element > max\_ele:

                max\_ele = element

        return max\_ele

the else statement contains a for loop used to iterate(moving through or using each number in the array one after the other) through the whole array(arr).It also has a **max\_ele** which contains the first number in the array arr[0].

    else:

        max\_ele = arr[0]

        for element in arr:

            if element > max\_ele:

                max\_ele = element

        return max\_ele

The for loop has an **element** which is used to store the variables in arr being looped through. The For loop contains an if statement that checks if the maximum element is the first number in the array, if it is then it stores the number in **max\_ele** then returns it (prints it on the screen), else it checks the next number after the second number to see if that’s the biggest number. It does this till it finds the largest number in the array.

        max\_ele = arr[0]

        for element in arr:

            if element > max\_ele:

                max\_ele = element

        return max\_ele

**Function3(count\_elemtment):**

This function is used to count the number of times a number repeats in an array.

def count\_element(arr, target):

    if not arr:

        return "An array with numbers is needed"

    else:

        count\_x = 0

        for x in arr:

            if x == target:

                count\_x += 1

        return count\_x

this function also has an if statement that is used to check if the array is a proper array or not.

if not arr:

        return "An array with numbers is needed"

the else part is the important part. It has a variable called **count\_x** which stores the number of times an element in the array occurs. But at the beginning of the it stores 0 as the value and it will be counting up from there.

count\_x = 0

In the For loop it is iterating through the array(arr), in this code the number we are checking if its being repeated is represented as **target.**  Now in the for loop there is an if statement that checks if an element in the array(arr) is equal to the **target,** and each times it finds a match the count\_x is increased by one:

count\_x += 1

this occurs till all the numbers in the array is exhausted

for x in arr:

            if x == target:

                count\_x += 1

        return count\_x

**Function 3(sorted):**

This function is used to check if an array is sorted or not

for i in range(len(arr) - 1):

        if arr[i] > arr[i + 1]:

            return False

    return True

In the code below it also uses a For loop to accomplish this. The for loop has a variable names **i** to store the numbers being iterated. Now this For loop uses the attribute **range** which is used to check the range or values or numbers to iterate up to. Now the attribute **len** is also used here, it’s to check the number of elements in the array. So with this code:

for i in range(len(arr) - 1):

we will iterate through a range of value from the length of the array -1. This is done so we don’t iterate passed the number of index we have. This will be explained further.

In the For loop there is an if statement. This if statement is used to check between two number which is the largest. It uses the argument (if arr[i] > arr[i+1]) in this argument, the element in the array is represented by i, hence we check the element **i** and the next element after it which is **i+1.** This is where the len(arr) -1 comes in, this statement helps use not to go over the index boundary. Since we are comparing the element in the array arr[i] and the next element arr[i+1] we can go overboard hence the statement **len(arr) -1** having the –1 added to it