Date: 06.03.2024 Name: Mugilkrishna D U
Ex. No: 2 Reg. No.: 3122225001073

UCS 2403 Design & Analysis of Algorithms

Assignment 2

Date of Exercise: 06.03.2024

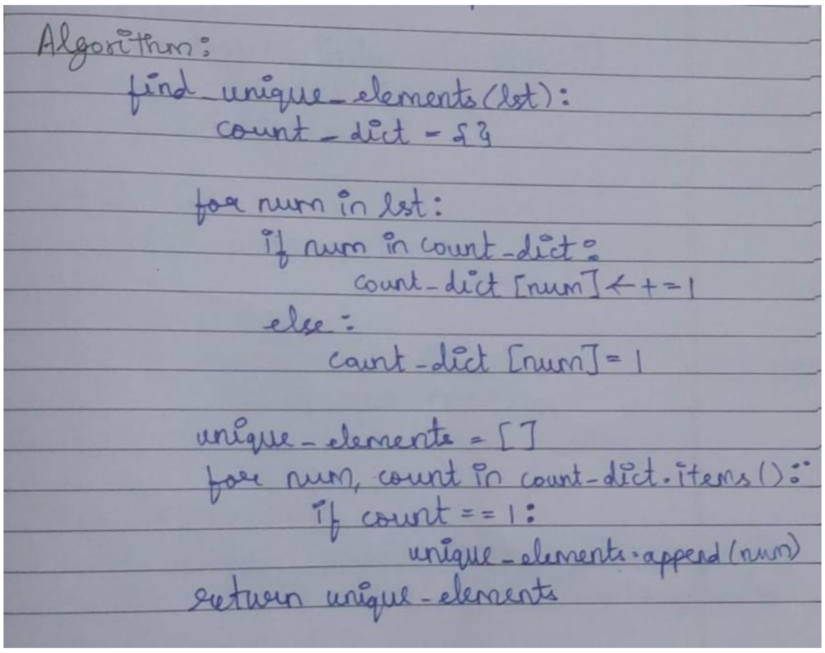
Aim: To gain understanding and proficiency in various iterative algorithms

Question 1:

(a) Develop a Python program to find unique (non-repeating) elements in a list. That is, find those elements that do not have duplicates in the list. For example, in the list [3, 6, 9, 2, 3, 9, 1, 15, 21, 3, 1], the unique elements are [6, 2, 15, 21]. The order of elements in the output list should be the same as that in the original list.

(b) What is the time complexity of your algorithm? You may ignore the improvements introduced by language specific implementations (say, using set in Python)

Algorithm:

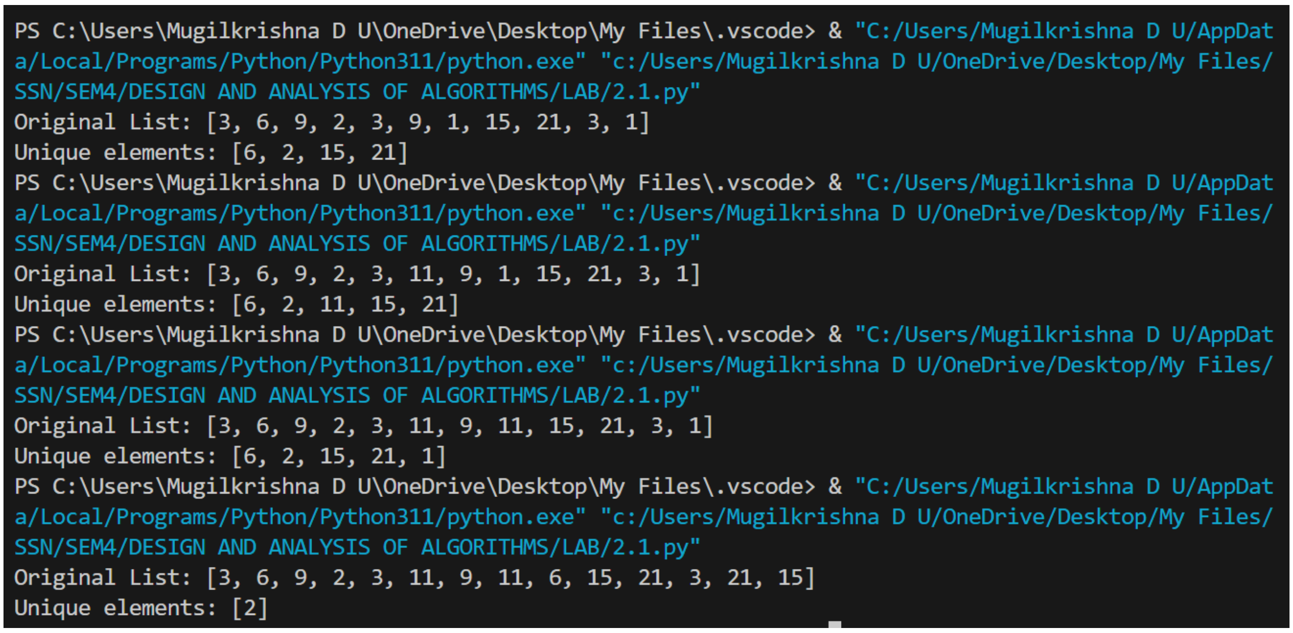


```
Algorithm:
    find_unique_elements(lst):
        count_dict = {}

        for num in lst:
            if num in count_dict:
                count_dict [num] ~~←~~ += 1
            else:
                count_dict [num] = 1

        unique_elements = []
        for num, count in count_dict.items():
            if count == 1:
                unique_elements.append(num)
        return unique_elements
```

Code:

```python
def find_unique_elements(lst):
    count_dict = {}
    # Counting occurrences of each element
    for num in lst:
        if num in count_dict:
            count_dict[num] += 1
        else:
            count_dict[num] = 1

    # List to store unique elements
    unique_elements = []

    # Adding unique elements to the list
    for num, count in count_dict.items():
        if count == 1:
            unique_elements.append(num)

    return unique_elements
```

Output:



```
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode> & "C:/Users/Mugilkrishna D U/AppDat
a/Local/Programs/Python/Python311/python.exe" "c:/Users/Mugilkrishna D U/OneDrive/Desktop/My Files/
SSN/SEM4/DESIGN AND ANALYSIS OF ALGORITHMS/LAB/2.1.py"
Original List: [3, 6, 9, 2, 3, 9, 1, 15, 21, 3, 1]
Unique elements: [6, 2, 15, 21]
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode> & "C:/Users/Mugilkrishna D U/AppDat
a/Local/Programs/Python/Python311/python.exe" "c:/Users/Mugilkrishna D U/OneDrive/Desktop/My Files/
SSN/SEM4/DESIGN AND ANALYSIS OF ALGORITHMS/LAB/2.1.py"
Original List: [3, 6, 9, 2, 3, 11, 9, 1, 15, 21, 3, 1]
Unique elements: [6, 2, 11, 15, 21]
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode> & "C:/Users/Mugilkrishna D U/AppDat
a/Local/Programs/Python/Python311/python.exe" "c:/Users/Mugilkrishna D U/OneDrive/Desktop/My Files/
SSN/SEM4/DESIGN AND ANALYSIS OF ALGORITHMS/LAB/2.1.py"
Original List: [3, 6, 9, 2, 3, 11, 9, 11, 15, 21, 3, 1]
Unique elements: [6, 2, 15, 21, 1]
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode> & "C:/Users/Mugilkrishna D U/AppDat
a/Local/Programs/Python/Python311/python.exe" "c:/Users/Mugilkrishna D U/OneDrive/Desktop/My Files/
SSN/SEM4/DESIGN AND ANALYSIS OF ALGORITHMS/LAB/2.1.py"
Original List: [3, 6, 9, 2, 3, 11, 9, 11, 6, 15, 21, 3, 21, 15]
Unique elements: [2]
```

Time Complexity: O(n) - Linear Time Complexity

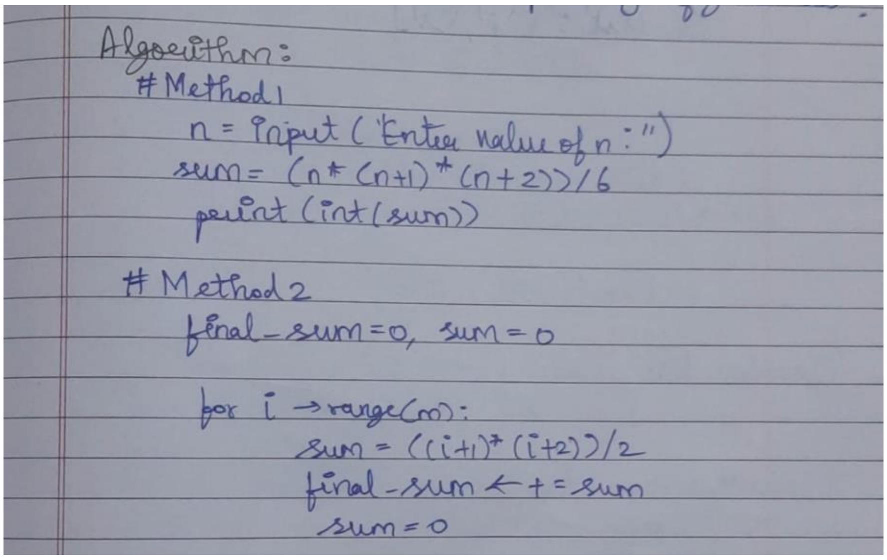
Question 2:

(a) Develop a Python program that when given an integer n as input, prints the sum of the following series up to n terms.
$1 + (1 + 2) + (1 + 2 + 3) + . . . + (1 + 2 + 3 + . . . + n)$

(b) What is the time complexity of your code?

Date: 06.03.2024 Name: Mugilkrishna D U
Ex. No: 2 Reg. No.: 3122225001073

Algorithm:

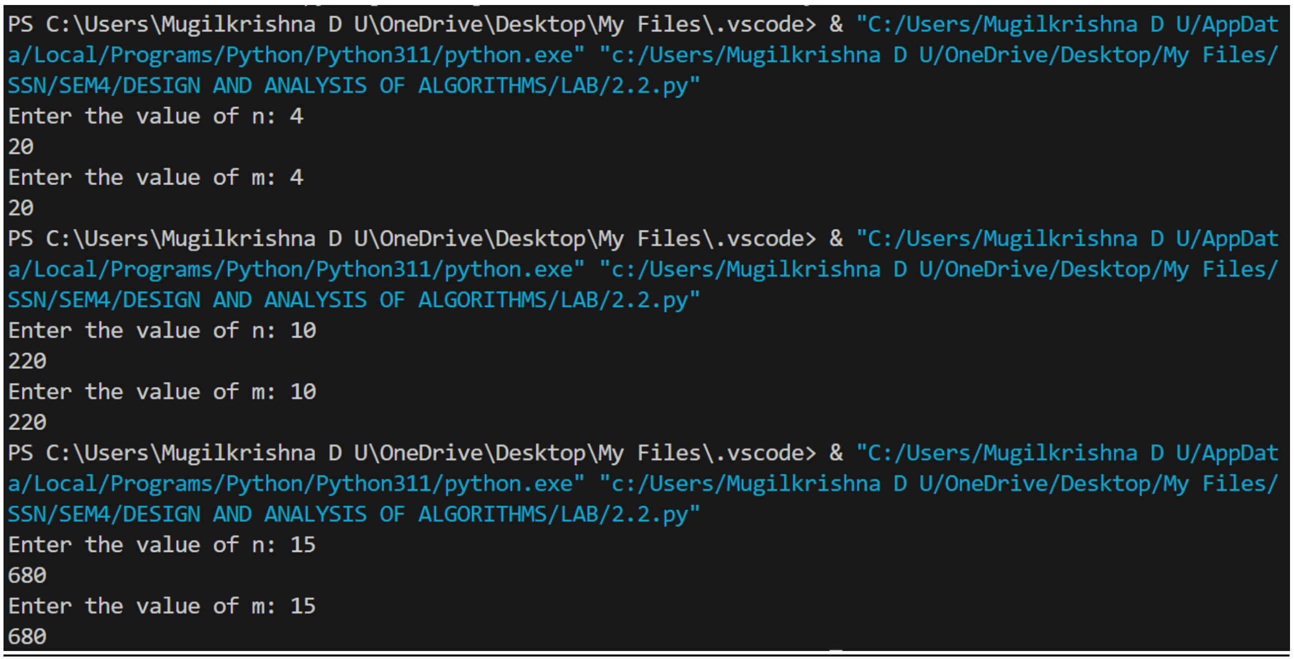


```
Algorithm:
 #Method1
    n = input ('Enter value of n:")
   sum = (n* (n+1)* (n+2))/6
    print (int(sum))

 #Method2
    final_sum=0, sum=0

    for i → range(m):
          sum = ((i+1)* (i+2))/2
          final_sum ← += sum
           sum=0
```

Code:

```
#Method 1

n = int(input("Enter the value of n: "))
sum = (n*(n+1)*(n+2))/6
print(int(sum))

#Method 2

m = int(input("Enter the value of m: "))
final_sum = 0
sum = 0

for i in range(m):
    sum = ((i+1)*(i+2))/2
    final_sum += sum
    sum = 0

print(int(final_sum))
```

Output:



```
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode> & "C:/Users/Mugilkrishna D U/AppDat
a/Local/Programs/Python/Python311/python.exe" "c:/Users/Mugilkrishna D U/OneDrive/Desktop/My Files/
SSN/SEM4/DESIGN AND ANALYSIS OF ALGORITHMS/LAB/2.2.py"
Enter the value of n: 4
20
Enter the value of m: 4
20
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode> & "C:/Users/Mugilkrishna D U/AppDat
a/Local/Programs/Python/Python311/python.exe" "c:/Users/Mugilkrishna D U/OneDrive/Desktop/My Files/
SSN/SEM4/DESIGN AND ANALYSIS OF ALGORITHMS/LAB/2.2.py"
Enter the value of n: 10
220
Enter the value of m: 10
220
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode> & "C:/Users/Mugilkrishna D U/AppDat
a/Local/Programs/Python/Python311/python.exe" "c:/Users/Mugilkrishna D U/OneDrive/Desktop/My Files/
SSN/SEM4/DESIGN AND ANALYSIS OF ALGORITHMS/LAB/2.2.py"
Enter the value of n: 15
680
Enter the value of m: 15
680
```

Time Complexity:

- **(i)** **Method 1:** O(1) – Constant Time Complexity
- **(ii)** **Method 2:** O(n) – Linear Time Complexity

Question 3:

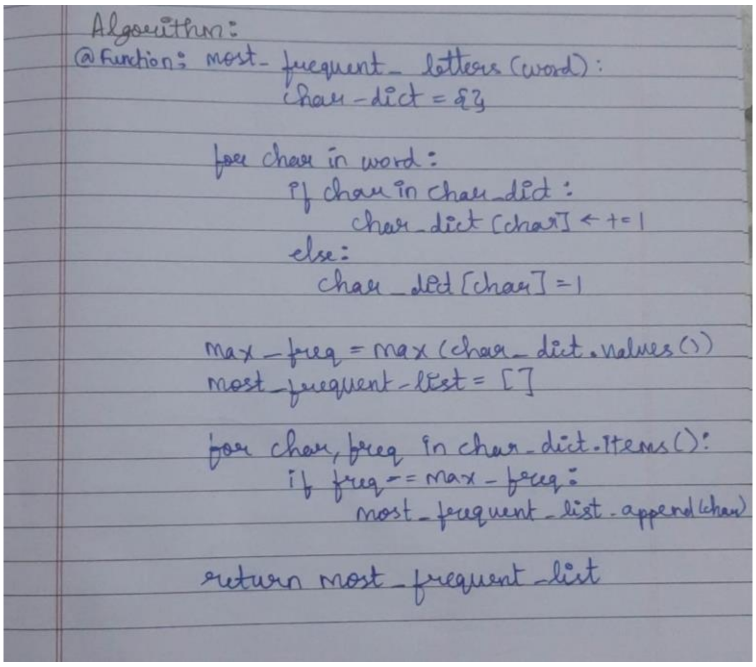
(a) Write a program to print all the most frequently occurring characters in a given string, as a list. For example, if the input string is “exam-ple”,the output should be [e]. If the input string is “exist”, then the output should be [e,x,i,s,t].

(b) What is the complexity of your code?

Date: 06.03.2024
Ex. No: 2

Name: Mugilkrishna D U
Reg. No.: 3122225001073

Algorithm:

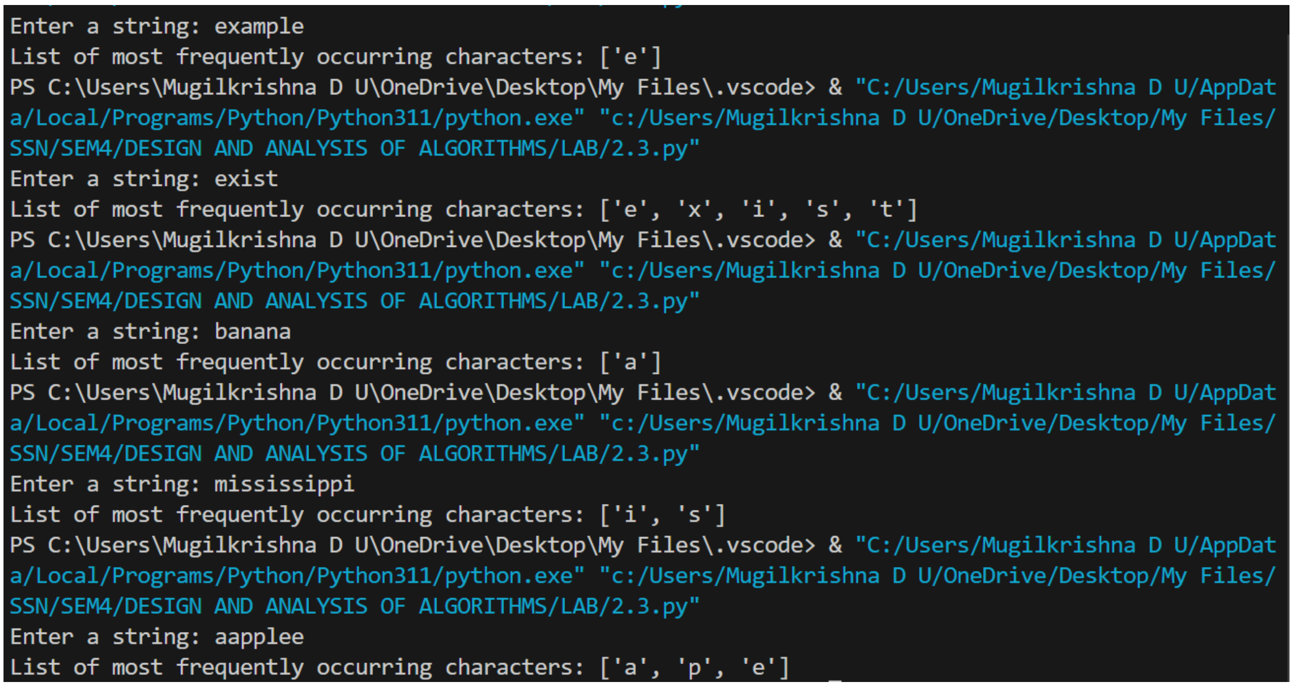


```
Algorithm:
@Function: most_frequent_letters (word):
    char_dict = {}

    for char in word:
        if char in char_dict:
            char_dict [char] ← +=1
        else:
            char_dict [char] =1

    max_freq = max (char_dict.values())
    most_frequent_list = []

    for char, freq in char_dict.items():
        if freq == max_freq:
            most_frequent_list.append(char)

    return most_frequent_list
```

Code:

```python
def most_frequent_letters(word):
    char_dict = {}

    # Count the occurrences of each character in the string
    for char in word:
        if char in char_dict:
            char_dict[char] += 1
        else:
            char_dict[char] = 1

    # Find the maximum frequency
    max_freq = max(char_dict.values())

    #Create a list to store the maximum frequency letters
    most_frequent_list = []

    for char, freq in char_dict.items():
        if freq == max_freq:
            most_frequent_list.append(char)

    return most_frequent_list

word = input("Enter a string: ")
frequency = most_frequent_letters(word)
print("List of most frequently occurring characters:", frequency)
```

Output:



```
Enter a string: example
List of most frequently occurring characters: ['e']
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode> & "C:/Users/Mugilkrishna D U/AppDat
a/Local/Programs/Python/Python311/python.exe" "c:/Users/Mugilkrishna D U/OneDrive/Desktop/My Files/
SSN/SEM4/DESIGN AND ANALYSIS OF ALGORITHMS/LAB/2.3.py"
Enter a string: exist
List of most frequently occurring characters: ['e', 'x', 'i', 's', 't']
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode> & "C:/Users/Mugilkrishna D U/AppDat
a/Local/Programs/Python/Python311/python.exe" "c:/Users/Mugilkrishna D U/OneDrive/Desktop/My Files/
SSN/SEM4/DESIGN AND ANALYSIS OF ALGORITHMS/LAB/2.3.py"
Enter a string: banana
List of most frequently occurring characters: ['a']
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode> & "C:/Users/Mugilkrishna D U/AppDat
a/Local/Programs/Python/Python311/python.exe" "c:/Users/Mugilkrishna D U/OneDrive/Desktop/My Files/
SSN/SEM4/DESIGN AND ANALYSIS OF ALGORITHMS/LAB/2.3.py"
Enter a string: mississippi
List of most frequently occurring characters: ['i', 's']
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode> & "C:/Users/Mugilkrishna D U/AppDat
a/Local/Programs/Python/Python311/python.exe" "c:/Users/Mugilkrishna D U/OneDrive/Desktop/My Files/
SSN/SEM4/DESIGN AND ANALYSIS OF ALGORITHMS/LAB/2.3.py"
Enter a string: aapplee
List of most frequently occurring characters: ['a', 'p', 'e']
```

Time Complexity: O(n) – Linear Time Complexity

Learning Outcome:
Upon completing this exercise, I have understood the importance of algorithmic efficiency and to analyse the time complexity of their solutions, which enables me to make informed decisions about algorithm selection and optimization.