CSE221: Algorithms
Lab Assignment 01
Fall 2022

Submission Guidelines:

1. You can code all of them either in Python or Java. But you should choose a specific language for all tasks.
2. For **Problem 2a**, write down the explanations in your copy, create a pdf file with the images of your hand written answers and name it **explanation2a.pdf**. For **problem 2b**, you need to download the image(.png) of the graph with a value of n good enough to differentiate between the two implementations. Name it **problem2b.png**.
3. Apart from problem 2, for each task write separate python files like task1.py, task3.py and so on.
4. For each problem, take input from files called "inputX.txt", and output at "outputX.txt", where X is the task number. So, for problem 2, the input file is basically this, "input2.txt". Same for output.
5. For each task include the input files (if any) in the submission folder.
6. Finally zip all the files and rename this zip file as per this format: **LabSectionNo_ID_CSE221LabAssignmentNo_Fall2022.zip**. [Example: **LabSection01_21101XXX_CSE221LabAssignment01_Fall2022.zip**]
7. You **MUST** follow all the guidelines, and naming/file/zipping convention stated above. **Failure to do so will result in a straight 50% mark deduction.**
8. Don't copy from your friends. **For plagiarism you will get 0**.

Task 1: File I/O

a) You are given a file **“input1a.txt”.** The first line of the input file will contain an integer **T**, representing the number of test cases. The next T lines will contain an integer **N**. You have to calculate if the number is Odd or Even. For each test case print the expected output. All the results must be compiled in a single file, **“output1a.txt**.**”**

Sample Input File	Sample Output File
5 10 19 7 3 100	10 is an Even number. 19 is an Odd number. 7 is an Odd number. 3 is an Odd number. 100 is an Even number.

b) You are given a file **“input1b.txt”.** The first line of the input file will contain an integer **T**, representing the number of test cases. The next T lines will contain a single arithmetic expression. Each arithmetic expression will start with the prefix “calculate“. It is guaranteed that the expression will have exactly two operands and one operator.

Calculate the result based on the operation (operator) on the operands and for each line of expression, write the result with “The result of $a $operator $b is $result” where $a and $b are the values of the first and second operator respectively, and $operator is the operator and $result is the result of the operation. All the results must be compiled in a single file, **“output1b.txt**.**”**

Sample Input File	Sample Output File
15 calculate 67 + 41 calculate 85 / 5 calculate 13 - 56 calculate 99 - 95 calculate 3 / 10 calculate 12 * 19 calculate 14 - 6 calculate 3 * 88 calculate 45 * 68 calculate 81 - 0 calculate 77 + 40 calculate 8 * 84 calculate 73 - 22 calculate 85 - 86 calculate 28 * 58	The result of 67 + 41 is 108 The result of 85 / 5 is 17.0 The result of 13 - 56 is -43 The result of 99 - 95 is 4 The result of 3 / 10 is 0.3 The result of 12 * 19 is 228 The result of 14 - 6 is 8 The result of 3 * 88 is 264 The result of 45 * 68 is 3060 The result of 81 - 0 is 81 The result of 77 + 40 is 117 The result of 8 * 84 is 672 The result of 73 - 22 is 51 The result of 85 - 86 is -1 The result of 28 * 58 is 1624

Task 2: Time Complexity Analysis

a) You are given two different codes for finding the n-th Fibonacci number. Find the time complexity of both implementations and compare the two.

Implementation 1

```
def fibonacci_1(n):
    if n < 0:
        print("Invalid input!")
    elif n <= 1:
        return n
    else:
        return fibonacci_1(n-1)+fibonacci_1(n-2)

n = int(input("Enter a number: "))
nth_fib = fibonacci_1(n)
print("The {} th fibonacci number is {}.".format(n, nth_fib))
```

Implementation 2

```
def fibonacci_2(n):
    if n<0:
        return "Invalid Input"
    if n<=1:
        return n
    fib = [0] * (n+1)
    fib[0] = 0
    fib[1] = 1
    for i in range(2,n+1):
        fib[i] = fib[i-1] + fib[i-2]
    return fib[n]

n = int(input("Enter a number: "))
nth_fib = fibonacci_2(n)
print("The {} th fibonacci number is {}.".format(n, nth_fib))
```

b) Append the following code segment after the implementations given in the previous problem. [Yes, The code is given. Just Copy-Paste it]. This will generate a graph with the value of **n** along the x-axis and the **time required** along the y-axis. You can see both curves in the same graph for better comparison. Generate graphs for different values of n and see how the performances change drastically for larger values of n.

```
import time
import math
import matplotlib.pyplot as plt
import numpy as np

#change the value of n for your own experimentation
n = 30
x = [i for i in range(n)]
y = [0 for i in range(n)]
z = [0 for i in range(n)]
for i in range(n-1):
  start = time.time()
  fibonacci_1(x[i+1])
  y[i+1]= time.time()-start
  start = time.time()
  fibonacci_2(x[i+1])
  z[i+1]= time.time()-start

x_interval = math.ceil(n/10)
plt.plot(x, y, 'r')
plt.plot(x, z, 'b')
plt.xticks(np.arange(min(x), max(x)+1, x_interval))
plt.xlabel('n-th position')
plt.ylabel('time')
plt.title('Comparing Time Complexity!')
plt.show()
```

Task 3:

Here is the code of bubble sort. Its run time complexity is $\theta(n^2)$. Change the code in a way so that its time complexity is $\theta(n)$ for the **best-case** scenario.

You have to **explain how you have achieved the** $\theta(n)$ for the **best-case** scenario **in a comment block of your code.**

```
def bubbleSort(arr):
    for i in range(len(arr)-1):
        for j in range(len(arr)-i-1):
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]
```

Input 1: 5 3 2 1 4 5	Output 1: 1 2 3 4 5
Input 2: 6 5 10 15 20 25 30	Output 2: 5 10 15 20 25 30

For the input 2, your code should run at θ(n).

Please note, you have to take the input from an input3.txt file, and show the output in an output3.txt file.

Task 4:

Suppose you are given a task to rank the students. You have gotten the marks and id of the students. Now your task is to rank the students based on their marks using a sorting algorithm.

However, you have to keep in mind that your sorting algorithms perform the minimum number of swapping operations.

Input:

The first line of the input file will contain an integer **N (1 ≤ N ≤ 1000)**.
The second line will contain N integers, representing the Student ID, Si (**1 ≤** Si **≤ 1000**).
The next line will contain the N integer, Sm (**1 ≤** Sm **≤ 1000**), which denotes the obtained mark of the corresponding students.

Output:

You have to show the Student Id and obtained marks in descending order based on their obtained mark. If two or more students get the same mark, then students with the lower ID will get prioritized. See the input and output for a better understanding.

Input 1: 7 7 4 9 3 2 5 1 40 50 50 20 10 10 10	Output 1: ID: 4 Mark: 50 ID: 9 Mark: 50 ID: 7 Mark: 40 ID: 3 Mark: 20 ID: 1 Mark: 10 ID: 2 Mark: 10 ID: 5 Mark: 10
Input 2: 4 7 2 5 3 80 60 80 50	Output 2: ID: 5 Mark: 80 ID: 7 Mark: 80 ID: 2 Mark: 60 ID: 3 Mark: 50

Please note, you have to take the input from an input4.txt file, and show the output in an output4.txt file.

Task 5:

You have been recently recruited as the Software Engineer at Jumanji Railway Software System. You have a big task at hand. You will be given the N **(1 ≤ N ≤ 100)** schedule of the train. The next N line will contain the name of the train and the departure time. See the input format for better understanding.

Your task is to write a sorting algorithm that will group the trains in the lexicographical order based on the name of the trains. If two or more trains have the same name, then the train with the latest departure time will get prioritized. If there is still a tie, then the train which comes first in the input file will come first.

Sample Input	Sample Output
13 ABCD will departure for Mymensingh at 00:30 DhumketuExpress will departure for Chittagong at 02:30 SubornoExpress will departure for Chittagong at 14:30 ABC will departure for Dhaka at 17:30 ShonarBangla will departure for Dhaka at 12:30 SubornoExpress will departure for Rajshahi at 14:30 ABCD will departure for Chittagong at 01:00 SubornoExpress will departure for Dhaka at 11:30 ABC will departure for Barisal at 03:00 PadmaExpress will departure for Chittagong at 20:30 ABC will departure for Khulna at 03:00 ABCE will departure for Sylhet at 23:05 PadmaExpress will departure for Dhaka at 19:30	ABC will departure for Dhaka at 17:30 ABC will departure for Barisal at 03:00 ABC will departure for Khulna at 03:00 ABCD will departure for Chittagong at 01:00 ABCD will departure for Mymensingh at 00:30 ABCE will departure for Sylhet at 23:05 DhumketuExpress will departure for Chittagong at 02:30 PadmaExpress will departure for Chittagong at 20:30 PadmaExpress will departure for Dhaka at 19:30 ShonarBangla will departure for Dhaka at 12:30 SubornoExpress will departure for Chittagong at 14:30 SubornoExpress will departure for Rajshahi at 14:30 SubornoExpress will departure for Dhaka at 11:30

Please note, you have to take the input from an input5.txt file, and show the output in an output5.txt file.