Problem Set - 2

Topic: Loops

Section 1: for, while, and 1D Lists

<table>
<tr><th>SL</th><th>Problem</th><th>Difficulty</th></tr>
<tr><td>1</td><td>Write a program to take a number as user input. Next, print all the numbers in ascending order up to that number (print the series up to Nth terms) in a single line.

<table><tr><th>Sample Input</th><th>Sample Output</th></tr><tr><td>2</td><td>1 2</td></tr><tr><td>5</td><td>1 2 3 4 5</td></tr><tr><td>11</td><td>1 2 3 4 5 6 7 8 9 10 11</td></tr></table></td><td>*</td></tr>
<tr><td>2</td><td>Write a program to take a number as user input. Next, print all the numbers in descending order up to that number (print the series up to Nth terms) in a single line.

<table><tr><th>Sample Input</th><th>Sample Output</th></tr><tr><td>2</td><td>2 1</td></tr><tr><td>11</td><td>11 10 9 8 7 6 5 4 3 2 1</td></tr></table></td><td>*</td></tr>
<tr><td>3</td><td>Write a program to take a number N as user input and generate a binary number in one line. The length of the number is N. (print 0s and 1s up to N).

<table><tr><th>Sample Input</th><th>Sample Output</th></tr><tr><td>2</td><td>10</td></tr><tr><td>5</td><td>10101</td></tr><tr><td>11</td><td>10101010101</td></tr></table>
[HINT: Find the pattern. Look at the indexes carefully. What pattern do they follow for each test case]</td><td>*</td></tr>
</table>

<table>
<tr>
<td>4</td>
<td>Write a program that will take the number N as user input. Next, the user will enter those N numbers and compute their average. Refer to here on how to take a list input in a single line.
<table>
<tr><th>Sample Input</th><th>Sample Output</th></tr>
<tr><td>3
10 20 30.5</td><td>Average: 20.1666….</td></tr>
<tr><td>2
22.4 11.1</td><td>Average: 16.7500….</td></tr>
</table>
</td>
<td>*</td>
</tr>
<tr>
<td>5</td>
<td>Write a program that takes two numbers x and y as user inputs. The inputs must be taken so that the value of y must be greater than the value of x (y > x). Next, the program will print the <u>square of all the numbers</u> starting from x all the way up to y. After reaching this range, the program will print “END”.
<table>
<tr><th>Sample Input</th><th>Sample Output</th></tr>
<tr><td>5 10</td><td>25 36 49 64 81 100 END</td></tr>
<tr><td>2 10</td><td>4 9 15 25 36 49 64 81 100 END</td></tr>
<tr><td>10 10</td><td>END</td></tr>
</table>
</td>
<td>*</td>
</tr>
<tr>
<td>6</td>
<td>Write a program that takes in two numbers as input. Next, find the GCD (greatest common divisor) and the LCM (lowest common multiple) of these two numbers.
<table>
<tr><th>Sample Input</th><th>Sample Output</th></tr>
<tr><td>36 48</td><td>GCD: 12
LCM: 144</td></tr>
<tr><td>5 10</td><td>GCD: 5
LCM: 10</td></tr>
</table>
</td>
<td>*</td>
</tr>
<tr>
<td>7</td>
<td>Write a program that takes a number N as input. Next, print the factorial of that number.
<table>
<tr><th>Sample Input</th><th>Sample Output</th></tr>
<tr><td>5</td><td>120</td></tr>
<tr><td>10</td><td>3628800</td></tr>
</table>
</td>
<td>*</td>
</tr>
</table>

<table>
<tr>
<td>8</td>
<td>Write a program that takes a number N as input. Next, check if the number is a prime number or not. Recall that a prime number is a number that can be divided by only 1 and itself.

<table>
<tr><th>Sample Input</th><th>Sample Output</th></tr>
<tr><td>2</td><td>Prime</td></tr>
<tr><td>6</td><td>Not Prime</td></tr>
<tr><td>1</td><td>Not Prime</td></tr>
<tr><td>11</td><td>Prime</td></tr>
</table>
</td>
<td>*</td>
</tr>
<tr>
<td>9</td>
<td>Write a program that will take the number N as input. Next, print the Fibonacci series up to the length of N.

<table>
<tr><th>Sample Input</th><th>Sample Output</th></tr>
<tr><td>1</td><td>1</td></tr>
<tr><td>2</td><td>1 1</td></tr>
<tr><td>4</td><td>1 1 2 3</td></tr>
<tr><td>7</td><td>1 1 2 3 5 8 13</td></tr>
</table>
</td>
<td>*</td>
</tr>
<tr>
<td>10</td>
<td>Write a program that will take the number N as input. Next, print the sum of the first Nth terms for the series:
1, -2, 3, -4, 5, -6, 7, -8, 9, -10, 11, -12, 13, -14, ...

<table>
<tr><th>Sample Input</th><th>Sample Output</th></tr>
<tr><td>2</td><td>3</td></tr>
<tr><td>10</td><td>11</td></tr>
<tr><td>0</td><td>1</td></tr>
</table>

Explanation

For the first example, the input is 2. Which means we need to find the 2nd index from the series. So 1 - (0th index), -2 - (1st index), 3 (2nd index)
</td>
<td>*</td>
</tr>
</table>

<table>
<tr>
<td>11</td>
<td>Write a program that takes a number and then prints the reverse of that number.

[Do this problem using the while statement]

[NOTE: Do not use Python’s shortcuts. You must strictly use the loops.]

<table>
<tr><th>Sample Input</th><th>Sample Output</th></tr>
<tr><td>12345678</td><td>87654321</td></tr>
<tr><td>900</td><td>009</td></tr>
<tr><td>3</td><td>3</td></tr>
</table>
</td>
<td>**</td>
</tr>
<tr>
<td>12</td>
<td>Write a program that will take a list as input. Refer to here on how to take a list input in a single line. Next, find out the maximum element present in that list.

<table>
<tr><th>Sample Input</th><th>Sample Output</th></tr>
<tr><td>1 2 3 4 5 600 9</td><td>600</td></tr>
<tr><td>9 8 0</td><td>9</td></tr>
<tr><td>1</td><td>1</td></tr>
</table>
</td>
<td>*</td>
</tr>
<tr>
<td>13</td>
<td>Write a program that will take a list as input. Next, find out, separately, the sum of the odd and even index numbers of the list. If the list has one value only, print “Cannot do this”.

<table>
<tr><th>Sample Input</th><th>Sample Output</th></tr>
<tr><td>1 2 3 4 5 6</td><td>Even index sum: 9
Odd index sum: 12</td></tr>
<tr><td>456</td><td>Cannot do this</td></tr>
</table>

Explanation

For the first example, the even index numbers are 1 3 5, which is 9. The odd index numbers are 2 4 6, which is 12.</td>
<td>**</td>
</tr>
</table>

<table>
<tr>
<td>14</td>
<td>You are working on a program to manage the roster of a sports team. The team's roster is represented as a list of jersey numbers, where each jersey number corresponds to a player. However, there seems to be an issue with duplicate jersey numbers, and you need to identify and resolve it.

Write a Python program that takes a list of jersey numbers as input and identifies and prints the jersey numbers that have duplicates. The printing of the jersey numbers in the output can be in any order.

<table>
<tr><th>Sample Input</th><th>Sample Output</th></tr>
<tr><td>3 7 2 8 2 5 7 1 8</td><td>Duplicate jersey numbers: 2 7 8</td></tr>
<tr><td>1 2 3 4 5</td><td>No duplicates</td></tr>
</table>
</td>
<td>**</td>
</tr>
<tr>
<td>15</td>
<td>You are developing a salary calculation program for XYZ FinTech LLC. The company pays its employees based on the number of hours worked and an hourly wage. However, there are certain rules for overtime pay:

Regular hours (up to 40 hours): Paid at the regular hourly wage.
Overtime hours (more than 40 hours): Paid at 1.5 times each hour more than the regular hourly wage.

The program should then print the total salary for each employee.

The program takes two lists as input: one containing the hours worked by each employee and the other containing their corresponding hourly wage. Calculate the hourly wage of each employee in separate lines.

<table>
<tr><th>Sample Input</th><th>Sample Output</th></tr>
<tr><td>Hours Worked: 38 45 30 50
Hourly Wage: 15 20 18 25</td><td>Employee 1: 570 BDT
Employee 2: 950 BDT
Employee 3: 540 BDT
Employee 4: 1375 BDT</td></tr>
</table>
</td>
<td>**</td>
</tr>
<tr>
<td>16</td>
<td>Design a program for your university's annual sports day, where traditional paper records are replaced by a sensor-based system to determine the winners of a 10m race. The program should accept participants' finish times as input, <u>with each index representing a participant number and the value at that index representing their finish time in seconds</u>. The program should then identify and display the participants who secured the first, second, and third positions. There can <u>never be two or more participants that finish at the same time</u>.

Next page</td>
<td>***</td>
</tr>
</table>

Sample Input	Sample Output
1 2 3 4 5	First Place: 0 Second Place: 1 Third Place: 2
5 1 2 4 9 10 7	First Place: 1 Second Place: 2 Third Place: 3
5 1 9	First Place: 1 Second Place: 0 Third Place: 2

Explanation:

For the second test case 5 1 2 4 9 10 7, the fastest time is 1 which is at index 1. So Participant 1 is first. The second fastest time is 2 at index 2, so Participant 2 is second. The third fastest time is 4 at index 3, so Participant 3 is third.

NEXT PAGE FOR THE SECOND SECTION

Section 2: Nested Loops, 2D Lists, and Patterns

<table>
<tr><th>SL.</th><th>Problem</th><th>Difficulty</th></tr>
<tr><td>17</td><td>Write a program to take a number N as input. Now, draw a right-angled triangle using *, up to the height N.

[NOTE: Use loops]

<table>
<tr><th>Sample Input</th><th>Sample Output</th></tr>
<tr><td>3</td><td>*
**
***</td></tr>
<tr><td>5</td><td>*
**
***
****
*****</td></tr>
<tr><td>100</td><td>(Not an output. You get the idea of how this should print)</td></tr>
</table>
</td><td>*</td></tr>
<tr><td>18</td><td>Write a program to take a number N as input. Now, draw an upside-down right-angled triangle using *

<table>
<tr><th>Sample Input</th><th>Sample Output</th></tr>
<tr><td>5</td><td>*****
****
***
**
*</td></tr>
</table>
</td><td>*</td></tr>
</table>

<table>
<tr>
<td>19</td>
<td>Write a program to take a number N as input. Now, draw a right-angled triangle that contains the squares of each row as their fill content (carefully check the output, and print spaces between each fill content).
<table>
<tr><th>Sample Input</th><th>Sample Output</th></tr>
<tr><td>3</td><td>1
44
999</td></tr>
<tr><td>5</td><td>1
4 4
9 9 9
16 16 16 16
25 25 25 25 25</td></tr>
</table>
</td>
<td>*</td>
</tr>
<tr>
<td>20</td>
<td>Write a program to print to take a number N as input. The program should print this diamond pattern with numbers.
<table>
<tr><th>Sample Input</th><th>Sample Output</th></tr>
<tr><td>3</td><td> 1
 212
32123
 212
  1</td></tr>
<tr><td>5</td><td>   1
   212
  32123
 4321234
543212345
 4321234
  32123
   212
    1</td></tr>
</table>
</td>
<td>**</td>
</tr>
<tr>
<td>21</td>
<td>Write a program to print to take a number N as input. The program should generate an identity matrix.
<table>
<tr><th>Sample Input</th><th>Sample Output</th></tr>
<tr><td>3</td><td>1 0 0
0 1 0
0 0 1</td></tr>
</table>
</td>
<td>*</td>
</tr>
</table>

	5	1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1
22	Write a program that would take two matrices **A and B** as inputs. For this question, assume that the dimension of matrix A is 3 x 2 and the matrix B is 2 x 3. Calculate the matrix multiplication of the two matrices in 2 x 2 dimensions. Look here on how to take a matrix input. *Next Page* **Sample Input** / **Sample Output** Enter Matrix A 1 2 3 4 5 6 Enter Matrix B 7 8 9 10 11 12 / 58 64 139 154	*
23	Write a program to take two numbers N and M. N represents the number of rows of the matrix, and M represents the number of columns. Next, take the matrix as input and find the maximum number present in each row of the matrix. **Sample Input** / **Sample Output** 3 3 1 2 3 1 4 9 76 34 21 / 3 9 76 5 5 1 2 3 21 12 1 65 9 1 56 34 2 / 21 65 56	*
24	Write a program to take two numbers N and M. N represents the number of rows of the matrix, and M represents the number of columns. Next, take the matrix as input and transpose the matrix. Recall the theory of matrix transpose here.	**

	<table><tr><th>Sample Input</th><th>Sample Output</th></tr><tr><td>1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 0</td><td>1 5 9 3 2 6 0 4 3 7 1 5 4 8 2 0</td></tr></table>	
25	Write a program that would take a matrix as input. Next, find the sum of the major (top left to bottom right) and minor (top right to bottom left) diagonals of the matrix. If the major diagonal sum is greater, then print **“Major aligned”**. If the minor diagonal sum is greater, the print **“Minor aligned”**. If they are the same, then print **“Balanced”**. <table><tr><th>Sample Input</th><th>Sample Output</th></tr><tr><td>9 0 1 1 2 1 3 3 3</td><td>Major sum: 14 Minor sum: 6 Major aligned</td></tr><tr><td>3 0 5 3 4 2 10 1 1</td><td>Major sum: 8 Minor sum: 19 Minor aligned</td></tr><tr><td>1 0 1 0 1 0 1 1 1</td><td>Major sum: 3 Minor sum: 3 Balanced</td></tr></table>	****