Programming I (Python)
Assignment 4

Instructions

- Similar to Assignment 2, 3.
- This assignment is about functions. **Please ensure that your code does not have any extraneous input/output code.**
- In several questions, underscores (`'_'`) have been used to highlight spaces (`' '`) in the output code. Your output should contain the space character (`' '`) in all those spaces.

Named Procedures

1. (a) Write a function `print_n_messages()` that prints "Hello world!" 10 times. (**file:** `Q1a.py`)
   (b) Write a function `print_n_messages(`n`)` that prints "Hello world!" n times. (**file:** `Q1b.py`)
   (c) Write a function `print_n_messages(`m`)` that prints message m 10 times. (**file:** `Q1c.py`)
   (d) Write a function `print_n_messages(`m`, `n`)` that prints message m n times. (**file:** `Q1d.py`)

2. (a) Write a function `banner(`m`)` that prints prints the message m decorated with borders. For example, `banner("Good Morning!")` with give:

```
*****************
*_Good Morning!_*
*****************
```

   (**file:** `Q2a.py`)

3. (a) Write a function `diamond()` that prints a diamond of height 5.

```
__*
_***
*****
_***
__*
```

   (**file:** `Q3a.py`)

(b) Write a function `diamond(n)` that prints a diamond of height n, where n is an odd number. Your function is not expected to behave deterministically if n is not an odd number. For example, `diamond(3)` will give:

```
_*
***
_*
```

and `diamond(5)` will give a diamond as printed in part(a). (**file:** `Q3b.py`)

(c) Write a function `diamond(n, c)` that prints a diamond of height n made of character c, where n is an odd number. Your function is not expected to behave deterministically if n is not an odd number. For example, `diamond(3, '1')` will give:

```
_1
111
_1
```

(**file:** `Q3c.py`)

(d) Write a function `diamond(n, c)` that prints a diamond of height n made of character c, where n is an odd number. Your function is not expected to behave deterministically if n is not an odd number. For example, `diamond(3, '1')` will give:

```
_1
111
_1
```

(**file:** `Q3d.py`)

4. Write a function `ndiamond()` that prints a numerical diamond of height 5.

```
__1
_121
12321
_121
__1
```

(**file:** `Q4.py`)

5. Write a function `ndiamond(n)` that prints a numerical diamond of height n. For example, `ndiamond(3)` will give:

```
_1
121
_1
```

and `ndiamond(5)` will give an output similar to the one in part (a). (**file:** `Q5.py`)

Mathematical Functions

6. Implement a function `hello(name)` that returns a string with "Hello " as prefix to name. (name is a string input.)

   Example:

```
$ python hello.py
Hello IIITB
```

7. Implement a function `double(l)` that takes an input list `l` and returns a list doubling every element of `l`. Use list comprehension to achieve this.

   Example:

```
$ python double.py
input =  [1, 2, 3]
[2, 4, 6]
```

8. Implement a function `even_elements(l)` that takes an input list `l` and returns a list only even elements from `l`. Use list comprehension to achieve this.

   Example:

```
$ python even.py
input =  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[0, 2, 4, 6, 8]
```

9. Write a function `balanced_brackets` that returns `True` if a given expression has balanced brackets; `False` otherwise. The input string is allowed to contain only three types of brackets: parentheses, i.e. `'('/')'`, curly braces `'{'/'}'` and square brackets `'['/']'`. Brackets of respective types must balance. Other characters are allowed but are ignored.

   (Hint: Implement a stack in Python using lists)

   (*Reference:* Balanced parentheses and stacks)

10. Write a function `transpose` that calculates (and returns) the transpose of the matrix m [1] passed to it as input parameter. (**file:** `Q10.py`)

11. Write a function `matmul` that calculates (and returns) the product of two matrices m_1 and m_2 passed to it as input parameters. Before beginning its main computation, `matmul` should check if m_1 and m_2 are multipliable. If they are not, `matmul` should return an appropriate message. (**file:** `Q11.py`)

[1] A matrix $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ is represented as a list of lists: `[[1, 2], [3, 4]]`. This holds for all instances of matrices in this assignment.