Programming I (Python)
Assignment 4

Instructions

- Answers to each question should be provided in a file whose name is mentioned against the respective question.
- 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.
- **How to submit.**
  1. The stub/starter files for all questions (except Q. 1) are provided in the directory named `answers`. Please write your answers in the files with appropriate names as given in the questions.
  2. Once you are satisfied with your solutions/answers, exit the `answers` directory and compress the `answers` directory preferably using the following command:

     ```
     tar cvzf answers.tar.gz answers
     ```

  3. Upload `answers.tar.gz` as the submission to the assignment.

Theory Questions

1. (a) Write a short note on side-effects in imperative style programming.
   (b) Distinguish between *named procedures* and *mathematical functions*. Give an example of each.
   (c) Justify the following statements: "Stored procedures can not exist without side-effects."

   (**file:** `Q1.pdf` / `Q1.doc`/ `Q1.docx`)

Named Procedures

2. (a) Write a function `print_n_messages()` that prints "Hello world!" 10 times. (**file:** `Q2a.py`)
   (b) Write a function `print_n_messages(n)` that prints "Hello world!" n times. (**file:** `Q2b.py`)
   (c) Write a function `print_n_messages(m)` that prints message m 10 times. (**file:** `Q2c.py`)
   (d) Write a function `print_n_messages(m, n)` that prints message m n times. (**file:** `Q2d.py`)

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

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

[1]

(**file:** `Q3.py`)

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

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

   (**file:** `Q4a.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:** `Q4b.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:** `Q4c.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] Note that `'_'` denotes a space character in the questions 3, 4, 5, 6.

```
_1
111
_1
```

(**file:** `Q4d.py`)

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

```
__1
_121
12321
_121
__1
```

(**file:** `Q5.py`)

6. 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:** `Q6.py`)

Mathematical Functions

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

   Example:

```
msg = Hello("IIITB")
print(msg)
```

   will print

```
hello IIITB
```

8. 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:

```
lst = double([1, 2, 3])
print(lst)
```

   will print

```
[2, 4, 6]
```

9. 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:

```
lst = even_elements([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
print(lst)
```

   will print

```
[0, 2, 4, 6, 8]
```

10. 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. Example:

```
print(balanced_brackets("[]"))
print(balanced_brackets("}{"))
```

   will print

```
True
False
```

   (Hint: Implement a stack in Python using lists)

   (*Reference:* Balanced parentheses and stacks)

11. Write a function `mattrans` that calculates (and returns) the transpose of the matrix m [2] passed to it as input parameter. (**file:** `Q11.py`) Example:

```
print(mattrans([[1, 2], [3, 4]]))
```

   will print

```
[[1, 3], [2, 4]]
```

12. 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:** `Q12.py`) Example:

```
print(matmul([[1, 2, 3], [4, 5, 6]], [[7, 10], [8, 11], [9, 12]]))
```

   will print

```
[[50, 68], [122, 67]]
```

[2] 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.