ENGG1330 Computer Programming

Tutorial 5

Objectives:

1. Write programs with flow control statements
2. Solve problems using functions

Tutorial participation (1%):

- t5_vpl_1
- Submission period: **within your OWN tutorial period**

Tutorial/take-home exercises (2%):

- Remaining problems in the worksheet
- Submission deadline: **noon, 8-MAR-2023 (Wednesday)**

t5_vpl_1. Print Patterns

Write a function to print each of the following patterns using nested loops. Note that all the rows in a pattern have the same length, filled with either a "#" character or a white space. There are four patterns, so, you need to write four functions. Below is the format of the function header.

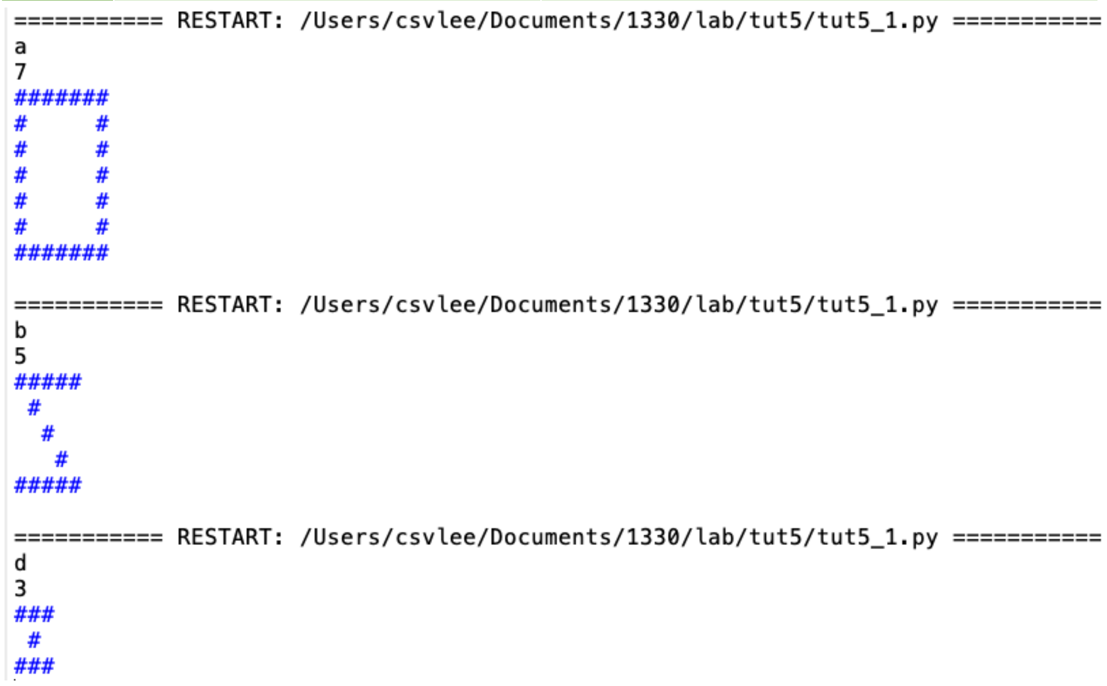
```
# 'X' is 'A', 'B', 'C' or 'D', size is a positive odd number.
def printPatternX(size):
```

```
# # # # # # #    # # # # # # #    # # # # # # #    # # # # # # #
#           #      #                        #        #       #
#           #        #                    #            #   #
#           #          #                #                #
#           #            #            #                #   #
#           #              #        #                #       #
# # # # # # #    # # # # # # #    # # # # # # #    # # # # # # #
     (a)               (b)             (c)              (d)
```

Write a program which accepts a character ('a', 'b', 'c' or 'd'), specifying the pattern to be printed, and an integer, the size of the pattern, and then calls the corresponding function to print the specified pattern. You may assume that the size is a positive odd number.

Sample cases and screenshots

Case	Input	Output
1	a 7	####### # # # # # # # # # # #######
2	b 5	##### # # # #####
3	d 3	### # ###



```
=========== RESTART: /Users/csvlee/Documents/1330/lab/tut5/tut5_1.py ===========
a
7
#######
#     #
#     #
#     #
#     #
#     #
#######

=========== RESTART: /Users/csvlee/Documents/1330/lab/tut5/tut5_1.py ===========
b
5
#####
 #
  #
   #
#####

=========== RESTART: /Users/csvlee/Documents/1330/lab/tut5/tut5_1.py ===========
d
3
###
 #
###
```

t5_vpl_2. Perfect Numbers

A positive integer is called a perfect number if the sum of all its factors (excluding the number itself, i.e., proper divisor) is equal to its value.

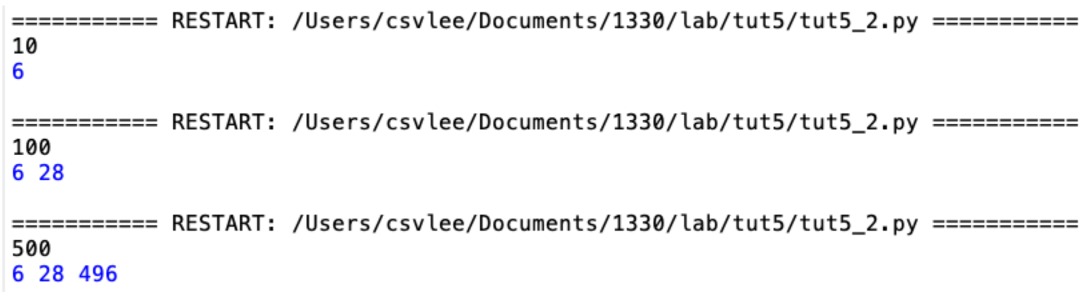
For example, the number 6 is perfect because its proper divisors are 1, 2, and 3, and 6=1+2+3; but the number 10 is not perfect because its proper divisors are 1, 2, and 5, and 10≠1+2+5.

Write a function called `isPerfect(posInt)` that takes a positive integer and returns `True` if the number is perfect.

Using the function, write a program that prompts user for an upper bound (a positive integer), and lists all the perfect numbers less than or equal to this upper bound.

Sample cases and screenshots

Case	Input	Output
1	10	6
2	100	6 28
3	500	6 28 496



```
=========== RESTART: /Users/csvlee/Documents/1330/lab/tut5/tut5_2.py ===========
10
6

=========== RESTART: /Users/csvlee/Documents/1330/lab/tut5/tut5_2.py ===========
100
6 28

=========== RESTART: /Users/csvlee/Documents/1330/lab/tut5/tut5_2.py ===========
500
6 28 496
```

t5_vpl_3. Palindrome

A palindrome is a word that reads the same backward as forward, e.g., noon, Madam. The following program reads a single word (a string of alphabets) from user, calls the method `lower()` to convert all the characters of the word to lower case, then calls the function `parse()` to parse the word into a parsed word and lastly calls another method `isupper()` to determine whether the parsed word is a palindrome or not by checking if all characters of the parsed word are in upper case.

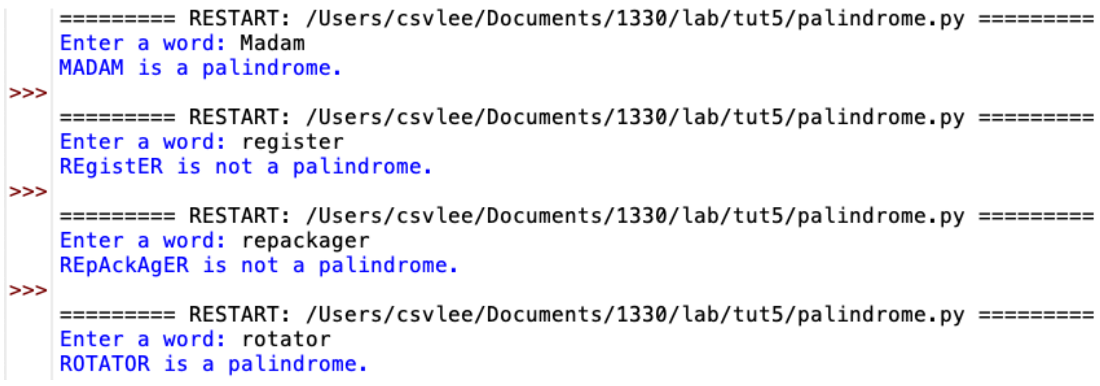
```
word=input('Enter a word: ')
parsed_word=parse(word.lower())
if parsed_word.isupper():
    print(f'{parsed_word} is a palindrome.')
else:
    print(f'{parsed_word} is not a palindrome.')
```

The function `parse()` takes the input word in lowercase, scans the word in both directions, capitalizes characters that are the same in the course of scanning and returns a parsed word.

Complete the program by writing the function `parse()`.

Sample cases and screenshots

Case	Input	Output
1	Madam	MADAM is a palindrome.
2	register	REgistER is not a palindrome.
3	repackager	REpAckAgER is not a palindrome.
4	rotator	ROTATOR is a palindrome.



```
========= RESTART: /Users/csvlee/Documents/1330/lab/tut5/palindrome.py =========
Enter a word: Madam
MADAM is a palindrome.
>>>
========= RESTART: /Users/csvlee/Documents/1330/lab/tut5/palindrome.py =========
Enter a word: register
REgistER is not a palindrome.
>>>
========= RESTART: /Users/csvlee/Documents/1330/lab/tut5/palindrome.py =========
Enter a word: repackager
REpAckAgER is not a palindrome.
>>>
========= RESTART: /Users/csvlee/Documents/1330/lab/tut5/palindrome.py =========
Enter a word: rotator
ROTATOR is a palindrome.
```

t5_vpl_4. Reverse sentence

Flipping a sentence means reversing the order of the words in the given sentence but the order of letters in a word does not change. For example, given a sentence of 'This is a dog.', the flipped sentence is 'Dog a is this.'. Note the following in the flipped sentence.

- The first letter of the first word has to be lowercased after flipping.
- The first letter of the last word has to be capitalized after flipping.
- The full-stop remains at the end of the sentence.

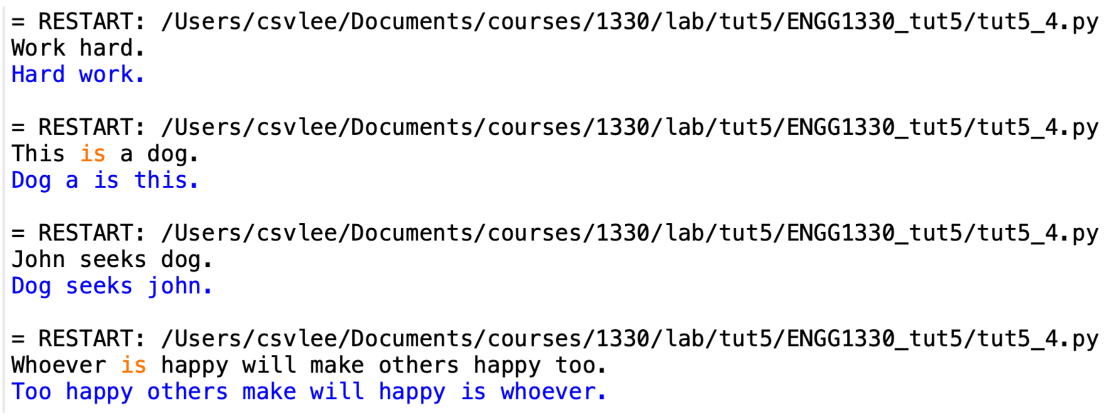
Write a function `reverse_sentence()` that takes a sentence and returns the flipped sentence. The input sentence is assumed a simple typical structure, starting with a capitalized letter, one space between words and no punctuation mark except a full-stop at the end of the sentence.

Using the function, write a program that prompts user for the sentence and prints the flipped sentence returned by the function.

You can use the method `lower()` to convert a character to lowercase, and use `upper()` to capitalize a character. e.g. `ch='a'`, then `ch.upper()='A'`; `ch='A'`, `ch.lower()= 'a'`. Also, if necessary, you can use the method `isalpha()` to determine whether a character is an alphabet or not.

Sample cases and screenshots

Case	Input	Output
1	`Work hard.`	`Hard work.`
2	`This is a dog.`	`Dog a is this.`
3	`John seeks dog.`	`Dog seeks john.`
4	`Whoever is happy will make others happy too.`	`Too happy others make will happy is whoever.`



```
= RESTART: /Users/csvlee/Documents/courses/1330/lab/tut5/ENGG1330_tut5/tut5_4.py
Work hard.
Hard work.

= RESTART: /Users/csvlee/Documents/courses/1330/lab/tut5/ENGG1330_tut5/tut5_4.py
This is a dog.
Dog a is this.

= RESTART: /Users/csvlee/Documents/courses/1330/lab/tut5/ENGG1330_tut5/tut5_4.py
John seeks dog.
Dog seeks john.

= RESTART: /Users/csvlee/Documents/courses/1330/lab/tut5/ENGG1330_tut5/tut5_4.py
Whoever is happy will make others happy too.
Too happy others make will happy is whoever.
```