```
"""
     ESAME SEMINARIO FONDAMENTI ALGORITMI PYTHON   28 Giugno 2018


    - NON CAMBIATE IL NOME DELLE FUNZIONI
        - SE VOLETE POTETE AGGIUNGERE FUNZIONI VOSTRE, MA DOVETE SEMPRE
          RICHIAMARLE DALLE FUNZIONI CHE VI HO SCRITTO IO

    - QUANDO SCRIVO 'RITORNA', LA FUNZIONE DEVE _RITORNARE_ DEI VALORI,
      NON STAMPARLI CON LA PRINT !!!!

    - COMUNQUE, PRINT ADDIZIONALI DI DEBUG SONO CONCESSE

    - NON RIASSEGNATE I PARAMETRI IN INPUT! TRADOTTO: NON SCRIVETE _MAI_
      QUALCOSA DEL GENERE !!!

          def f(x):
             x = .....

    - OGNI FUNZIONE E' SEGUITA DAGLI ASSERT DI TEST, SE QUANDO ESEGUITE
      IL CODICE NON VEDETE ERRORI PROBABILMENTE IL CODICE E' GIUSTO
      (MA NON FIDATEVI TROPPO, I TEST NON SONO ESAUSTIVI !)

    - L'ESAME E' OPEN BOOK, POTETE TROVARE LA DOCUMENTAZIONE NELLA CARTELLA
      'Seminario Python' SUL DESKTOP. PER NAVIGARLA, CLICCATE SUL FILE index.html
      ALL'INTERNO. TROVERETE:

      - documentazione Python
      - Tutorial Nicola Casella
      - libro Pensare in Python
      - fogli algoritmi
            - per aprirli in Jupyter, copiateli sul desktop, aprite Juptyter e
              navigate fino a fogli col browser di Jupyter


    - ATTENTI AI WHILE INFINITI, TENDONO AD INCHIODARE I COMPUTER DEL LABORATORIO !
      SE POSSIBILE, USATE CICLI FOR !

    - USARE EDITOR SPYDER.
      - Per eseguire tutto lo script: F5
      - Per eseguire solo la linea corrente o la selezione:  F9

    *** DA FARE: IMPLEMENTATE QUESTE 4 FUNZIONI: ***

    - doppie
    - nostop
    - dilist
    - matrot

"""


"""
Prende in input una lista con n numeri interi, e RITORNA una NUOVA lista che
contiene n tuple ciascuna da due elementi. Ogni tupla contiene un numero
preso dalla corrispondente posizione della lista di partenza, e il suo doppio.

Per esempio:

doppie([ 5, 3, 8])

deve dare la nuova lista

[(5,10), (3,6), (8,16)]

"""
def doppie(lista):
    # scrivi qui


# INIZIO TEST - NON TOCCARE !
assert doppie([]) == []
assert doppie([3]) == [(3,6)]
assert doppie([2,7]) == [(2,4),(7,14)]
assert doppie([5,3,8]) == [(5,10), (3,6), (8,16)]
# FINE TEST

"""
Quando si analizza una frase, può essere utile processarla per rimuovere parole molto comuni, come
 gli articoli e le preposizioni:

Per esempio:

      "un libro su Python per principianti"

  si può semplificare in

      "libro Python principianti"

Le parole 'poco utili' vengono chiamate 'stopwords'. Questo processo è per esempio eseguito
dai motori di ricerca per ridurre la complessità della stringa di input fornita dall'utente.
```

```
Implementare una funzione che prende una stringa e RITORNA la stringa di input senza le
stopwords

SUGGERIMENTO 1: le stringhe in Python sono *immutabili* ! Per rimuovere le parole dovete
creare una _nuova_ stringa a partire dalla stringa di partenza.

SUGGERIMENTO 2: create una lista di parole così:
    lista = stringa.split(" ")

SUGGERIMENTO 3: operate le opportune trasformazioni su lista, e poi costruite la stringa
da restituire con  " ".join(lista)
"""
def nostop(stringa, stopwords):
    # scrivi qui


# INIZIO TEST - NON TOCCARE !
assert nostop("un", ["un"]) == ""
assert nostop("un", []) == "un"
assert nostop("", []) == ""
assert nostop("", ["un"]) == ""
assert nostop("un libro", ["un"]) == "libro"
assert nostop("un libro su Python", ["un","su"]) == "libro Python"
assert nostop("un libro su Python per principianti", ["un","uno","il","su","per"]) == "libro Python
principianti"
# FINE TEST

"""
Restituisce un dizionario con n coppie chiave-valore, dove le chiavi sono
numeri interi da 1 a n incluso, e ad ogni chiave i è associata una lista di numeri
da 1 a i

NOTA: le chiavi sono *numeri interi*, NON stringhe !!!!!

Esempio:

dilist(3)

deve dare:

 {
    1:[1],
    2:[1,2],
    3:[1,2,3]
}


"""
def dilist(n):
    # scrivi qui


# INIZIO TEST - NON TOCCARE !

assert dilist(0) == dict()
assert dilist(1) == {
                        1:[1]
                    }
assert dilist(2) == {
                        1:[1],
                        2:[1,2]
                    }
assert dilist(3) == {
                        1:[1],
                        2:[1,2],
                        3:[1,2,3]
                    }
# FINE TEST



import numpy as np

"""
RESTITUISCE una NUOVA matrice numpy che ha i numeri della matrice numpy
di input ruotati di una colonna.

Per ruotati intendiamo che:
- se un numero nella matrice di input si trova alla colonna j, nella matrice
di output si troverà alla colonna j+1 nella stessa riga.
- Se un numero si trova nell'ultima colonna, nella matrice di output
si troverà nella colonna zeresima.

Esempio:

Se abbiamo come input

np.array(   [
                [0,1,0],
                [1,1,0],
```

```
                [0,0,0],
                [0,1,1]
            ])

Ci aspettiamo come output

np.array(   [
                [0,0,1],
                [0,1,1],
                [0,0,0],
                [1,0,1]
            ])

"""
def matrot(matrice):
    # scrivi qui


# INIZIO TEST - NON TOCCARE !

m1 = np.array(  [
                    [1]
                ])
mat_attesa1 = np.array( [
                            [1]
                        ])

assert np.allclose(matrot(m1), mat_attesa1)

m2 = np.array(  [
                    [0,1]
                ])
mat_attesa2 = np.array( [
                            [1,0]
                        ])
assert np.allclose(matrot(m2), mat_attesa2)

m3 = np.array(  [
                    [0,1,0]
                ])
mat_attesa3 = np.array(
                [
                    [0,0,1]
                ])

assert np.allclose(matrot(m3), mat_attesa3)

m4 = np.array(  [
                    [0,1,0],
                    [1,1,0]
                ])
mat_attesa4 = np.array( [
                            [0,0,1],
                            [0,1,1]
                        ])
assert np.allclose(matrot(m4), mat_attesa4)


m5 = np.array([
                [0,1,0],
                [1,1,0],
                [0,0,0],
                [0,1,1]
              ])
mat_attesa5 = np.array([
                        [0,0,1],
                        [0,1,1],
                        [0,0,0],
                        [1,0,1]
                       ])
assert np.allclose(matrot(m5), mat_attesa5)
# FINE TEST
```