Student Name:	**Student ID:** **Section:** **Exam Room:**		**Total Grade:**
	Exam: Midterm	**Q1: 15**	**Q2: 30**
Acad. Year: 2019 - 2020 Fall	**Date: 16/11/2019**	**Q3: 30**	**Q4: 25**
Ins: Aynur Dayanık **İpek Sözen** **Lori Russell Dağ**	**Duration: 100 min.**		

In programming questions, answers considered too MESSY or OTHERWISE UNREADABLE will not be graded.

GOOD LUCK!

```
int() / float() / str()
input( message )
print( string, end = string )
range( start, stop, step )
```

String
```
len( obj )
format( value_list )
find( string )
index( element )
strip()
split( delimiter )
lower()
```

List
```
len( obj )
append( element )
extend( sequence )
insert( index, element )
pop( index )
remove( element )
index( element )
reverse()
count()
sort()
sorted( list )
```

Dictionary
```
pop( key )
keys()
```

Files
```
open( filename, mode )
write( string )
read()
readline()
readlines()
close()
```

Question 1 (15 points)

Write a function, **`reverse_dictionary()`** that takes a dictionary as a parameter and creates and returns a new dictionary where the keys and values from the input dictionary are swapped. I.e the keys become the values and the values become the keys.

Important Notes:

- You may assume that all **values** are unique in the dictionary.
- You may assume that all **values** are of immutable types in the dictionary.
- The input dictionary should NOT be modified.

For example: **`reverse_dictionary(`**`{'a':1, 'b':2}`**`)`** `returns {1:'a', 2:'b'}`

```
def reverse_dictionary( dict1 ):
    """Assumes dict1 is a dictionary.
    Returns a NEW dictionary where the values
    are the keys and the keys are the values, from the input
    dictionary
    """

    rev = {}
    for key in dict1:
        rev[dict1[key]] = key
    return rev
```

Question 2 **(30 points = 5 + 7 + 10 + 8 points)**

What is the output when the following programs are executed?

a)

```
a = [3,2,3,4,0]
i = 0
while i < len(a):
    a[a[i]] = i
    i += 1

print(a)
```

Answer:

[4, 2, 1, 0, 0]

b)

```
y = 3
for m in range(y):
    for n in range(y):
        print('m =',m,'n =',n)
        y = 1
```

Answer:

m = 0 n = 0
m = 0 n = 1
m = 0 n = 2
m = 1 n = 0
m = 2 n = 0

c)

```
def guess( e, fcn=int ):
   return fcn(e)

fcn1 = bool
fcn2 = abs
lst = [-81.78,0 ,25.6, -14.6, 38.75]
for i in range(len(lst)):
    if lst[i] > 20:
        lst[i] = guess(lst[i])
    elif lst[i] > -20:
        lst[i] = guess(lst[i], fcn1)
    else:
        lst[i] = guess(lst[i], fcn2)

print(lst)
```

Answer:

[81.78, False, 25, True, 38]

d)

```
tup = ('de', (1, 2, 3), 6, ['a','b','c'])

e1 = tup[1][1:2]
e1 += e1

e2 = tup[3][1:2]
e2 += e2

tup[3].extend(['x','y'])

print(tup)
print(e1, type(e1))
print(e2, type(e2))
```

Answer:

('de', (1, 2, 3), 6, ['a', 'b', 'c', 'x', 'y'])
(2, 2) <class 'tuple'>
['b', 'b'] <class 'list'>

Question 3 (30 points)

a) Write a function **`isDelectable()`** that takes a positive integer n and checks if n is a delectable number. A number is called *delectable* if the number formed by its first n digits is always divisible by n. For example, 4236 is delectable because

1. the number formed by the first digit (4) is divisible by 1,
2. the number formed by the first 2 digits (42) is divisible by 2,
3. the number formed by the first 3 digits (423) is divisible by 3,
4. the number formed by the first 4 digits (4236) is divisible by 4.

2052 is not delectable because, although 2 is divisible by 1, and 20 is divisible by 2, 205 is not divisible by 3.

Hint: you may use strings in your solution.

```
def isDelectable(n):
    """Assumes n is an int > 0
    Returns True if n is a delectable number,
    Returns False otherwise"""



    numStr = str(n)
    print(n)
    for i in range(2, len(numStr)+1):
        prefix = numStr[:i]
        num = int(prefix)
        if num % i  != 0:
            return False
    return True
```

b) Using the **`isDelectable`** function from part a), write a function named **`separateDelectables`** that takes a list of ints, L, and removes the delectable numbers from L and returns a new list containing the delectable numbers from L. Returns an empty list if there are no delectables in the input list.
Note: 0 points will be given if the function does not use the **isDelectable** function from part a

```
def separateDelectables(L):
    """Assumes L is a list of positive ints
    Removes delectable ints from L and
    Returns those delectable ints in a new list"""



    delectables = []
    i = 0
    while i < len(L):
        if isDelectable(L[i]):
            delectables.append(L.pop(i))
        else:
            i += 1
    return delectables
```

Question 4 (25 points)

Write a function, `distances()` that takes a `filename(str)`, an `origin(str)` and a `destination (str)` as parameters. The function should read the file, and starting from the `origin`, find and return the distance (kms) between the `origin` and the `destination`. If there is no route found between the two cities, the function should return -1.

Important Notes:

- Each line of the file contains `start_city`, `end_city` and the `distance` between the `start_city` and `end_city` (in kms).
- You may assume that the `start_city` on a given line was the `end_city` from the previous line (i.e. the next city on the route).
- The `origin` may not be the first city in the file, and the `destination` may not be the last city.
- You should NOT use a nested loop in your solution.

```
distances('distances.txt', 'izmir', 'ankara')   -> 592.0
distances('distances.txt', 'IZMIR', 'ankara')   -> 592.0
distances('distances.txt', 'ankara', 'ordu')    -> 558.6
distances('distances.txt', 'ankara', 'ORDU')    -> 558.6
distances('distances.txt', 'izmir', 'trabzon')  -> 1328.4
distances('distances.txt', 'izmir', 'antalya')  -> -1
distances('distances.txt', 'ankara', 'polatli') -> -1
distances('distances.txt', 'trabzon', 'izmir')  -> -1
distances('distances.txt', 'izmit', 'ankara')   -> -1
```

Sample **distances.txt** file contents (assume cities are ordered by route):

```
Izmir,Usak,220
Usak,Polatli,292
Polatli,Ankara,80
Ankara,Kirikkale,78.6
Kirikkale,Corum,164
Corum,Samsun,168
Samsun,Ordu,148
Ordu,Giresun,47.8
Giresun,Trabzon,130
```

***NOTE: Sample data only, file contents may change!**

WRITE YOUR ANSWER IN THE SPACE GIVEN ON THE NEXT PAGE.

```
def distance( filename, origin, destination):
    """Assumes filename, origin and destination
    are str values. Returns the distance between
    the origin and destination if route exists,
    returns -1 if no route found.
    """
    in_file = open(filename, 'r')
    #assume no route found
    calc = -1
    for line in in_file:
        dist = line.split(',')
        #if current line starts at origin
        if dist[0].lower() == origin.lower():
            #start calculation at zero
            calc = 0
        #if origin found (calculation not -1)
        if calc != -1:
            #add distance to calculation
            calc += float(dist[2])
            #if current line is at destination
            if dist[1].lower() == destination.lower():
                #exit, route found.
                break
    #if destination not found (last line did not contain destination)
    if dist[1].lower() != destination.lower():
        calc = -1
    return calc
```