

PROBLEM 1 (25 points):

Write a function called **buildDict()** which takes three lists as parameters. The first two lists are strings which, together, are a key into a dictionary. The values for the dictionary are in the third parameter. The lists are correlated by position and are the same size in length. If any of the lists are empty, you will return a list with a single value of -1. The key must be a tuple.

FOR FULL POINTS:

- Key must be a tuple of two with the city name in the first position and the state in the second
- Values must be stored with the correct key
- Code must be logically efficient (any code in there must be required for the program to work)
- Must be able to retrieve correct values based on key tuple
- Variables must be appropriately named (no 'x','y' but d, k, citylst, stlst, etc)
- Only one return used in the program

```
>>> d=buildDict(['Chicago','Detroit','New York','Seattle'], ['IL','MI','NY','WA'], ['a102','a44','b33','c3'])
>>> d[('Seattle', 'WA')]
'c3'
>>> d[('Chicago', 'IL')]
'a102'
```

PROBLEM 2 (25 points):

Write a function called **used()** which takes two lists as parameters. Each list is a mix of numeric and alphabetic characters. Each element in the lists represent the numbers from 0 to 9 and there are no values greater than 9. **Using a dictionary**, change any digits to string representations of the numbers. Combine the two lists and return a sorted list. **YOU MUST USE A DICTIONARY.**

FOR FULL POINTS:

- Must use a dictionary
- Code must be logically efficient (any code in there must be required for the program to work)
- Variables must be appropriately named (no 'x','y' but names which identify the variable)
- Results must include all members of the two lists
- Only one return used in the program

```
>>> useD(['one','two','three','4','5','6','7'], ['0','zero'])
['five', 'four', 'one', 'seven', 'six', 'three', 'two', 'zero', 'zero']
>>> useD(['two','three','4','five','6','3'], ['0','one','four'])
['five', 'four', 'four', 'one', 'six', 'three', 'three', 'two', 'zero']
>>> useD(['two','three','4','five','6','3'], [])
['five', 'four', 'six', 'three', 'three', 'two']
>>> useD([], ['1'])
['one']
>>> useD([], [])
[]
```

PROBLEM 3 (25 points):

Write a function called `sWork()` which takes a file and counts the number of occurrences in the file for each unique word. It returns a sorted list of tuples consisting of the count followed by the word.

FOR FULL POINTS:

- Must return a list of tuples
- Tuples consist of two parts: a count and the word.
- Sorted order is ascending by count
- Code must be logically efficient (any code in there must be required for the program to work)
- Variables must be appropriately named (no 'x','y' but names which identify the variable)
- Only one return used in the program

```
>>> sWork('stvdnp.txt')
[(1, '15'), (1, '1581'), (1, 'age'), (1, 'an'), (1, 'avoid'), (1, 'been'), (1, 'believed'),
(1, 'bernard'), (1, 'bertrande'), (1, 'birth'), (1, 'born'), (1, 'brothers'), (1, 'by'),
(1, 'child'), (1, 'childhood'), (1, 'claudine'), (1, 'correspondents'), (1, 'depaul'), (1,
'derivation'), (1, 'did'), (1, 'during'), (1, 'early'), (1, 'family'), (1, "family's"), (1
, 'family's'), (1, 'farmers'), (1, 'france'), (1, 'gayon'), (1, 'guyenne'), (1, 'had'), (1
, 'have'), (1, 'herder'), (1, 'him'), (1, 'inference'), (1, 'is'), (1, 'kingdom'), (1, 'li
vestock'), (1, 'managing'), (1, 'might'), (1, 'moras'), (1, 'mother'), (1, 'named'), (1, '
noble'), (1, 'none'), (1, 'one'), (1, 'oxen'), (1, 'pay'), (1, 'peasant'), (1, 'possibly')
, (1, 'pouy'), (1, 'province'), (1, 'reading'), (1, 'selling'), (1, 'seminary'), (1, 'sent
'), (1, 'showed'), (1, 'sisters'), (1, 'so'), (1, 'stream'), (1, 'talent'), (1, 'there'),
(1, 'third'), (1, 'this'), (1, 'three'), (1, 'two'), (1, 'vicinity'), (1, 'village'), (1,
'vincent'), (1, 'word'), (1, 'work'), (1, 'writing'), (1, 'wrote'), (2, 'as'), (2, 'at'),
(2, 'but'), (2, 'de'), (2, 'father'), (2, 'for'), (2, 'gascony'), (2, 'it'), (2, 'jean'),
(2, 'marie'), (2, 'name'), (2, 'paul'), (2, 'that'), (3, 'a'), (3, '-'), (4, 'to'), (5, 'h
e'), (5, 'his'), (5, 'in'), (5, 'was'), (7, 'and'), (7, 'of'), (11, 'the')]
```

```
>>> sWork('stvdnp2.txt')
[(1, '1605'), (1, 'administer'), (1, 'alchemist'), (1, 'an'), (1, 'arts'), (1, 'as'), (1,
'auctioned'), (1, 'back'), (1, 'barbary'), (1, 'became'), (1, 'bidder'), (1, 'bondage'), (
1, 'brought'), (1, 'but'), (1, 'captive'), (1, 'castres'), (1, 'de'), (1, 'due'), (1, 'fas
cinated'), (1, 'first'), (1, 'fisherman'), (1, 'for'), (1, 'gone'), (1, 'highest'), (1, 'h
im'), (1, 'how'), (1, 'inheritance'), (1, 'inventor'), (1, 'line'), (1, 'marseilles'), (1,
'master's'), (1, 'next'), (1, 'of'), (1, 'off'), (1, 'on'), (1, 'patron'), (1, 'paul'), (1
, 'physician'), (1, 'pirates'), (1, 'prepare'), (1, 'property'), (1, 'received'), (1, 'rem
edies'), (1, 'sailed'), (1, 'sea'), (1, 'sell'), (1, 'sickness'), (1, 'slave'), (1, 'sold'
), (1, 'some'), (1, 'soon'), (1, 'spagyric'), (1, 'spagyrical'), (1, 'spent'), (1, 'taken'
), (1, 'taught'), (1, 'the'), (1, 'this'), (1, 'toulouse'), (1, 'tunis'), (1, 'two'), (1,
'unsuitable'), (1, 'way'), (1, 'wealthy'), (1, 'where'), (1, 'who'), (1, 'work'), (1, 'yea
rs'), (2, 'by'), (2, 'had'), (2, 'master'), (2, 'vincent'), (3, 'from'), (3, 'he'), (4, 'a
'), (4, 'in'), (5, 'his'), (5, 'to'), (6, 'and'), (7, 'was')]
```

PROBLEM 4 (25 points):

Write a program called **jump()** which takes a two-dimensional list and a starting point into the list as parameters. The first entry in the sublist is a string and the second is a number. The number is to be used as an index into the list. The program moves through the list using the second entries in the sublist until the program is revisiting a sublist in the list. An exception is thrown if the starting point is less than 0 or greater than the length of the list. In this case, a list with a value of [] is returned. Use a try and except to catch the invalid points. The program will return a list which consists of the first entries in the sublists it has visited and the number of places visited appended to the end of the list.

FOR FULL POINTS:

- Must return a list consisting of the jumps IN ORDER taken and the total number of jumps taken
- Must use a try/except to check for both invalid start points
- Code must be logically efficient (any code in there must be required for the program to work)
- Variables must be appropriately named (no 'x','y' but names which identify the variable)
- Only one return used in the program

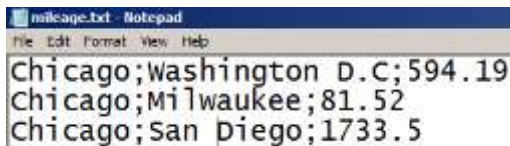
```
>>> lst=[['a',8],['b',2],['c',7],['d',1],['e',5],['f',0],['g',5],['h',8],['i',1]]
>>> jump(lst,2)
['c', 'h', 'i', 'b', 4]
>>> jump(lst,11)
[]
>>> jump(lst,-1)
[]
>>> jump(lst,8)
['i', 'b', 'c', 'h', 4]
>>> jump(lst,3)
['d', 'b', 'c', 'h', 'i', 5]
```

PROBLEM 5 (25 points):

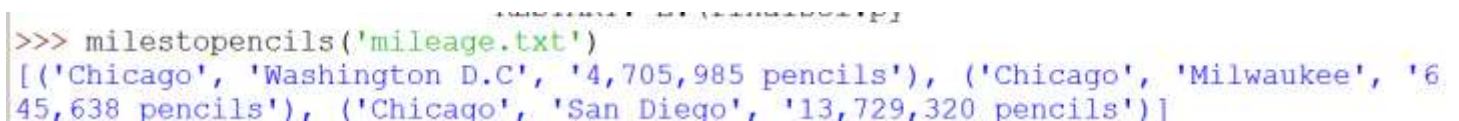
Write a program to calculate the distance between two points in terms of pencils. An average pencil is 8" in length. There are 63,360 inches in a mile. The parameter for this program is a file which has three columns. The first two are cities. The third is the distance between them in miles. Calculate the distance in pencils and create a tuple with both cities and the distance in pencils. Return the list.

FOR FULL POINTS:

- Number of pencils must be shown with "," in appropriate places and as whole number
- Result of each conversion must be in a tuple of three elements
- Code must be logically efficient (any code in there must be required for the program to work)
- Variables must be appropriately named (no 'x','y' but names which identify the variable)
- Only one return used in the program



```
mileage.txt - Notepad
File Edit Format View Help
Chicago;Washington D.C;594.19
Chicago;Milwaukee;81.52
Chicago;San Diego;1733.5
```



```
>>> milestopencils('mileage.txt')
[('Chicago', 'Washington D.C', '4,705,985 pencils'), ('Chicago', 'Milwaukee', '645,638 pencils'), ('Chicago', 'San Diego', '13,729,320 pencils')]
```

PROBLEM 6 (25 points):

Write a program to calculate the distance up and back between two points in terms of strides. The average woman's stride is 53". There are 63,360 inches in a mile. Determine the number of female strides it would take to go to and return from the two cities given. The parameter for this program is a file which has three columns. The first two are cities. The third is the distance between them in miles. Calculate the distance in strides giving a total from start to return. You must use try/except to identify invalid values for mileage (less than or equal to 0, non-numeric string) and missing city entries and return an error message as shown below. Output from the program must use **format** to format the error message and to format the strides results. Errors should be placed in a list as well as valid results. See below for examples. The program returns a list.

FOR FULL POINTS:

- Number of strides must be shown with "," in appropriate places and as whole number
- Result of each conversion must be in a list of three elements and list of these returned
- Code must be logically efficient (any code in there must be required for the program to work)
- Variables must be appropriately named (no 'x','y' but names which identify the variable)
- Only one return used in the program
- Output must use **format** error messages

```
mileage1 - Notepad
File Edit Format View Help
Chicago;Austin;0
Chicago;Washington D.C;594.19
Chicago;Milwaukee;81.52
Chicago;;99
Chicago;Detroit;ttt
Chicago;;
;;
Chicago;Richmond;-9
Chicago;San Diego;1733.5
```

```
>>> milestostrides('mileage1.txt')
['ERROR: Invalid input found in this line: Chicago;Austin;0', ['Chicago', 'Washington D.C', '1,420,675 strides'], ['Chicago', 'Milwaukee', '194,910 strides'], 'ERROR: Invalid input found in this line: Chicago;;99', 'ERROR: Invalid input found in this line: Chicago;Detroit;ttt', 'ERROR: Invalid input found in this line: Chicago;;', 'ERROR: Invalid input found in this line: ;;', 'ERROR: Invalid input found in this line: Chicago;Richmond;-9', ['Chicago', 'San Diego', '4,144,700 strides']]
```

```
mileage - Notepad
File Edit Format View Help
Chicago;Washington D.C;594.19
Chicago;Milwaukee;81.52
Chicago;San Diego;1733.5
```

```
>>> milestostrides('mileage.txt')
[['Chicago', 'Washington D.C', '1,420,675 strides'], ['Chicago', 'Milwaukee', '194,910 strides'],
 ['Chicago', 'San Diego', '4,144,700 strides']]
```

PROBLEM 7 (25 points):

Write a program to calculate the total sum of numeric digits found in an alphanumeric string. Input to this program will be a string and a number. The number will indicate where to start the calculations in the string. Digits in the string will be a single digit (0-9). If a "*" is found in the string, all calculations stop and the total at that time returned. If no numbers are found in the string, the n entered is greater than the length of the string, or the string is empty, a -1 is returned. A try/except must be used to handle the error conditions. (NOTE: Python numbering is used so n can be from 0 to one less than the string's length.)

FOR FULL POINTS:

- An integer must be returned
- Try/except must be used to handle errors in input
- Code must be logically efficient (any code in there must be required for the program to work)
- Variables must be appropriately named (no 'x','y' but names which identify the variable)
- Only one return used in the program

```
>>> addstring('abcdefg',3)
-1
>>> addstring('1abcd1efg',3)
1
>>> addstring('1abcd*1efg',3)
0
>>> addstring('1abcd*1efg',0)
1
>>> addstring('1abcd1efg*',0)
2
>>> addstring('',0)
-1
>>> addstring('',55)
-1
>>> addstring('12345*6789',0)
15
>>> addstring('12345*6789',6)
30
```

PROBLEM 8 (25 points):

Write a program **changestring()** will take as parameters three strings. The third string will replace occurrences of the second string in the first string up to the character '*' Changes will only occur up to the '*'. The program will return a string with the changes. If there is no "*" in the string, the entire string is used to make the change.

FOR FULL POINTS:

- Must use a try/except to check for empty first string and empty second string
- Code must be logically efficient (any code in there must be required for the program to work)
- Variables must be appropriately named (no 'x','y' but names which identify the variable)
- Only one return used in the program
- Must use format for the error message

```
>>> changestring('abc123def456abc','abc','xxx')
'xxx123def456xxx'
>>> changestring('abc123def456abc*','abc','xxx')
'xxx123def456xxx'
>>> changestring('ab*c123def456abc','abc','xxx')
'abc not present in ab*c123def456abc'
>>> changestring('abc*','abc','xxx')
'xxx'
>>> changestring('abcdefgh*','', 'xxx')
''
>>> changestring('','abc','xxx')
''
```