

PROBLEM 1 : (*What are the types and values? (24 points)*)

Consider the following variables and their values for the table below.

```
words = ['mint', 'orange', 'grape', 'cherry', 'all']
phrase = "test of space all 123"
```

List in the table the type of variable and its value after being assigned the expression.

variable = expression	Type	Value
a = "otter"[2]	string	't'
b = len(words[3])		
c = phrase[3] + phrase[-2]		
d = words[0]		
e = phrase.find('o')		
f = phrase.split("s")[1]		
g = words[2][3:]		
h = "cat".join(['2', 'dog'])		
i = "-".join(list("ciao"))		
j = len(phrase.split()) > len(words)		
k = words[2:4]		
m = 1.3 + 5/2		
n = 11 % 5		

PROBLEM 2 : (Simple Functions (14 points))

PART A: Discount (6 pts)

At the Super Discount Store, an item is discounted based on how many days it has been on sale, by multiplying the number of days times the percent for one day. If an item has been on sale 3 days, and the percentage discount is 10 percent per day, then 30 percent discount is taken off the regular price. If the percent off is more than 100 percent, the item has no cost. It is free, so return a cost of 0.

Write the function **calculate** that has three parameters: **price** is a float representing the regular cost of the item when there is no sale, **days** is an integer representing the number of days the item has been on sale, and **percent** is an integer representing the percentage off for one day. This function returns the sale price of the item as a decimal number.

call	returns	comment
calculate(100.00, 7, 10)	30.0	70 percent off the regular price
calculate(10.00, 3, 10)	7.0	30 percent off the regular price
calculate(52.00, 12, 10)	0.0	120 percent off the regular price

```
def calculate(price, days, percent):
```

PART B: How far (8 pts)

A football field is 100 yards long but the yards are marked from 0 to 50 yards, with 50 being the center and then counting back down from 50 to 0. We will call one side of the field the "A" side of the field and the other side the "B" side of the field. One football play goes from one location on the field to another location on the field. Your task is to compute the yardage for the one play. For example, here is the field showing every fifth yard:

"A" side | "B" side
0 5 10 15 20 25 30 35 40 45 50 45 40 35 30 25 20 15 10 5 0

Write the function **yardage** that has four parameters: **stfield** is a string indicating which side of the field the ball starts (either "A" or "B"), **styard** is an integer from 0 to 50 indicating the yard line the ball starts on, **efield** is a string indicating the side of the field the ball ends at, and **eyard** is an integer indicating the yard line the ball is on at the end of the play. This function returns total yardage for the play.

```
def yardage(stfield, styard, efield, eyard):
```

call	returns	comment
yardage("A", 24, "A", 48)	24	from A24 to A48, 24 on A side
yardage("A", 29, "B", 35)	36	from A29 to B35, 21 on A side and 15 on B side
yardage("B", 30, "A", 45)	25	from B30 to A45, 20 on B side and 5 on A side

PROBLEM 3 : *(It's a mystery and debugging (12 points))*

PART A: Mystery (6 pts)

Consider the following mystery function that has one parameter, where `words` is a list of strings. The lines have been numbered.

```

1 def mystery(words):
2     ans = []
3     for w in words:
4         if w[0] < 'm':
5             ans = [w] + ans
6         else:
7             ans.append(w)
8         # print ans
9     return ans

```

Q1. Consider the call to `mystery`.

```

wordlist = ["cow", "gerbil", "raccoon", "pig", "horse"]
result = mystery(wordlist)

```

For this sample call, what is the value assigned to `result`?

Q2. Suppose the print statement on line 8 is uncommented. For the sample call in Q1, what is the value printed the **first** time this print statement is executed?

Q3. Suppose the print statement on line 8 is uncommented. For the sample call in Q1, what is the value printed the **second** time this print statement is executed?

Q4. Explain in words what this function does, that is, what does it calculate for any given inputs?

PART B: Debugging (6 points)

Consider the attempt to implement the function **findNthVowel** that has two parameters: **word** is a string and **num** is an integer. This function is suppose to return the location of the **num**th vowel in word. If there are not at least num vowels in word, the function should return -1. **This function does not work correctly!**

Note that **findNthVowel** calls the function **isVowel** that has one string parameter **ch**. The function **isVowel(ch)** returns True if ch is a vowel and False if it is not. You can assume that isVowel works as intended.

```
1 def findNthVowel(word, num):
2     count = 0
3     for ch in word:
4         if count == num:
5             return word.find(ch)
6         if isVowel(ch):
7             count += 1
8     return -1
```

Here are two calls to **findNthVowel**, one with a correct answer and one with a wrong answer.

call	returns	correct answer	comment
findNthVowel("incredible", 5)	-1	-1	only 4 vowels
findNthVowel("minimalist", 3)	6	5	wrong position returned

Q1. Explain why the second call above is returning the wrong answer, 6.

Q2. Give an example of another word that returns the wrong answer

Q3. Give an example of a word that has at least **num** vowels and returns the correct answer.

Q4. Explain how to correct the code above so it always returns the intended answer. (you can mark your answer on the code above or just explain here).

PROBLEM 4 : (*Transformation (8 points)*)

Write the function **reformat** which has two parameters, **name** is a string of words separated by blanks that represent a person's name, and **num** is an integer. The parameter num is the number of words in the last name. You can assume there are always at least two words in name, the first name is one word, and num is greater than 0. This function returns name in a different format, with all the last names first (separated by blank), followed by a comma and a blank, followed by the first name, followed by a blank and followed by a string of the first letter of all the middle names. If there are no middle names, the name ends with the first name.

Consider these examples.

call	returns
reformat("Fred Smith", 1)	Smith, Fred
reformat('Mary Elizabeth Sue Bell Smith Johnson', 2)	Smith Johnson, Mary ESB
reformat('Xiaobai Yin Yu Xu Zhu Wei', 3)	Xu Zhu Wei, Xiaobai YY

```
def reformat(name, num):
```

PROBLEM 5 : (*Snacks before dinner (28 points)*)

Consider information about snacks that is stored in a file in the following format. Each line represents information about one snack. For each line there are four pieces of information and several separators (in this order): the name of the snack (one or more words), a blank, a left parenthesis, the weight of the snack, a right parenthesis, the number of calories as an integer, a space, and the type of snack (one or more words).

Shown below is a sample file. In the first line the snack is "Twix", it weighs 1.02 ounces, it has 144 calories and it is of type "chocolate".

```
Twix (1.02)144 chocolate
apple (6.4)95 fruit
avocado (5.29)240 fruit
stick pretzels (1.5)160 chips
orange (6.34)85 fruit
honey mustard pretzels (2.25)320 chips
m&ms (1.69)230 chocolate
peach (6.17)69 fruit
chex mix (1.72)210 chips
m&ms peanut (1.74)250 chocolate
mini pretzels (1.5)160 chips
KitKat (1.5)210 chocolate
Jack Links Beef Jerky (2.5)200 salty protein
pear (5.22)86 fruit
Skinny Pop popcorn (0.65)100 chips
hershey with almonds (1.45)210 chocolate
swedish fish (1.41)140 candy
banana (4.16)105 fruit
CheezIt (1.5)210 chips
raisins (5.82)493 fruit
Reeses peanut butter cups (1.50)210 chocolate
```

A. (6 pts)

Write the function **getParts** that has one parameter, a string in the format of one line from the previous page representing one snack, and **returns a list of four items** of information about that snack: the name of the snack as a string, the number of ounces as a float, the number of calories as an integer and the type of snack as a string.

```
def getParts(line):
```

B. (4 pts)

call	returns
getParts("Twix (1.02)144 chocolate")	["Twix", 1.02, 144, "chocolate"]
getParts("stick pretzels (1.5)160 chips ")	["stick pretzels", 1.5, 160, "chips"]

Consider the function named `fileToList` that has one string parameter `filename` that represents a datafile in the format on page 8 and returns a list of lists in the format shown below, where each list in the big list has four pieces of information representing one line from the data file (and thus one snack): the name of the snack as a string, the number of ounces as a float, the number of calories as an integer and the type of snack as a string.

The line, `datalist = fileToList("snackdata.txt")` where `snackdata.txt` is the file on page 8 results in the list of lists (not all shown):

```
datalist = [ ['Twix', 1.02, 144, 'chocolate'],
             ['apple', 6.4, 95, 'fruit'],
             ['avocado', 5.29, 240, 'fruit'],
             ...
             ['Reeses peanut butter cups', 1.5, 210, 'chocolate'] ]
```

The function is started below and has some missing lines indicated by MISSING LINES. The function should read in all the lines from the file and return the lists of lists in the format described above. **You must call the function `getParts` from part A in this function.** Add in the missing lines.

```
def fileToList(filename):
    answer = []
    f = open(filename)
    #MISSING LINES

    return answer
```

C. (8 points) Write the function `categories` that has one parameter named `snklist`, representing the list of lists in the format described in Part B, where each list has four items.

This function returns the **unique** list of types of snacks.

Assume `datalist` is the example list of lists formed in Part B from the data file on page 8. Then the call `categories(snklist)` returns the list `['chocolate', 'fruit', 'chips', 'salty protein', 'candy']`.

```
def categories(snklist):
```

D. (10 points) Write the function `fewestCaloriesPerOz` which has one parameter named `snklist` where `snklist` is a list of lists in the format described in Part B. (where each list is four items representing one snack). This function returns the **name of the snack** that has the fewest calories per ounce. If there is a tie, then return any one of those that tied. Consider the example below. Assume `datalist` is the example list of lists formed in Part B from the data file on page 8.

call	returns
<code>fewestCaloriesPerOz(datalist)</code>	"peach"

```
def fewestCaloriesPerOz(snklist):
```