Introduction to Computer Programming
Extra Practice Problems – Midterm #2

1. You have been hired by a computer retailer to build an online “shopping cart” application that lets customers design their own computer systems. Your employer sells computers based on the following pricing structure:

- Dell computers have a base price of $400
- HP computers have a base price of $600
- Users can select how much memory they want based on the following chart:

  4 GB = Included in the base price
  8 GB = $100
  16 GB = $200

- Users can also select a hard drive size based on the following chart:

  256 GB = Included in the base price
  512 GB = $50
  1024 GB = $100

Write a ***FUNCTION*** that accepts three arguments (a model (Dell or HP), amount of memory and hard drive size). Compute the cost of the desired computer using the pricing chart above and ***RETURN*** the result. Note that your function should be able to handle models (i.e. we don’t sell Macs) or invalid sizes for memory or hard drives – in these cases you should return an “Invalid Setup” string. Comment your function using IPO notation. Here’s a sample program that uses your function:

```
print( build_a_computer("Dell", 4, 256) ) # 400
print( build_a_computer("HP",   4, 256) ) # 600
print( build_a_computer("Mac",  4, 256) ) # Invalid Setup
print( build_a_computer("HP",   8, 256) ) # 700
print( build_a_computer("HP",   8, 512) ) # 750
print( build_a_computer("Dell", 4, 999) ) # Invalid Setup
```

Note: you are not writing a full program for this question -you will do that in the next question. Just write the function as specified.

2. *This question builds on Question #2. If you did not complete question #2 that's OK – you can assume you have access to a function called "build_a_computer" that works as specified in Question #2.*

Write a program that continually prompts the user to enter in a computer model, amount of memory and hard drive size. Next, check the cost of that setup using the function you wrote for the previous question. If the setup is valid you should add the cost of the setup to the user's cart and provide them with a running total of their bill. Here's a sample running of your program:

```
Your shopping cart is currently: 0 dollars
Enter a model (Dell or HP): HP
How much memory do you want? (4, 8, 16): 8
Hard drive size? (256, 512, 1024): 512
Your setup will cost: 750
Would you like to price another computer? yes

Your shopping cart is currently: 750 dollars
Enter a model (Dell or HP): Mac
How much memory do you want? (4, 8, 16): 4
Hard drive size? (256, 512, 1024): 256
We're sorry, but that setup is not available. Please try
again.
Would you like to price another computer? yes

Your shopping cart is currently: 750 dollars
Enter a model (Dell or HP): Dell
How much memory do you want? (4, 8, 16): 16
Hard drive size? (256, 512, 1024): 1024
Your setup will cost: 700
Would you like to price another computer? no

Your total shopping cart is: 1450 dollars
```

3. For this program you will be writing a program that scores a multiple-choice test. The answers to this particular multiple-choice test are expressed as a String. For example, the following defines the answers to a 28 question test:

```
key   = "ABABCDABCADABDABACCABDABCADA"
```

A student's individual answers are also expressed as a String. For example:

```
test  = "ABABCDCCCADABDABCBACADDBCADA"
```

If a student answered a question correctly they get a point. If not they do not get a point. For example, the key and test above would generate the following output:

```
Student earned 20 out of 28 possible points
```

Note that your program should work for any String, not just the one included in this problem. If you have too many or too few student answers you should report this and not score the test. For example:

```
key   = "ABABCDABCADABDABACCABDABCADA"
test  = "ABADABDADBCADDCCADDBDDADBDDADBBCAD"
```

```
Too many or too few answers!
```

You can always assume that you will have access to two Strings for the purpose of this program (key and test) – you do not need to prompt the user for this information.

You cannot assume that the test being scored will always have 28 answers – your program should work for tests with any number of answers.

4. A "checksum" is a computer programming technique that can be used to quickly compare two files to determine if they contain the same values. You can implement a simple checksum by adding up the ASCII Values of all elements within a String to form a single integer that represents the sum of all characters used in the String.

For this program you should write a **FUNCTION** called "checksum" that accepts a single argument (a String). The function should then **RETURN** the checksum back to the caller. Document your function using IPO notation.

Your main program should then prompt the user for two Strings and compare them by length and checksum. If both the length and checksum of the Strings are the same you should congratulate the user, and if not you should tell them that their Strings did not validate. Ask the user if they want to re-run the program once you are finished comparing the two Strings. Here's a sample running of the program:

```
Enter phrase 1: The cute dog
Enter phrase 2: Foobar
P1 has a length of 12 and a checksum of 1100
P2 has a length of 6 and a checksum of 601
Failed
Would you like to try again?  Yes

Enter phrase 1: The cute dog
Enter phrase 2: The dog cute
P1 has a length of 12 and a checksum of 1100
P2 has a length of 12 and a checksum of 1100
Passed!
Would you like to try again?  Yes

Enter phrase 1: python is cool
Enter phrase 2: school in typo
P1 has a length of 14 and a checksum of 1387
P2 has a length of 14 and a checksum of 1387
Passed!
Would you like to try again?  No
```

5. For this program you will be writing a ***FUNCTION*** that can be used to swap letter case within a String (i.e. swap all uppercase letters with their lowercase equivalents and vice-versa). Your function should accept ***ONE ARGUMENT*** – a String to analyze. Your function should then ***RETURN*** a copy of the newly created String. You cannot use any String methods, including “swapcase”, “isalpha”, “isupper” or “islower to solve this problem. Document your function using IPO notation.

Here are some sample calls to your function. Note that your function should behave exactly like the sample calls below.

```
print ( swapcase("Hello World") )   # hELLO wORLD
print ( swapcase("foobar")      )   # FOOBAR
print ( swapcase("123 456 789") )   # 123 456 789
print ( swapcase("")            )   # (no output)
```

Python Command Index

Core Language Elements and Functions	**Module Functions**
and	random.randint
chr	random.random
def	random.uniform
del	
elif	**String Testing Methods**
else	isalpha()
except	isdigit()
float	islower()
for	isupper()
format	isspace()
global	isalnum()
if	find()
import	
in	**String Modification Methods**
input	rstrip()
int	lstrip()
max	lower()
min	upper()
not	capitalize()
open	title()
or	swapcase()
ord	replace()
print	
range	
return	
str	
str.lower	
str.upper	
try	
while	

ASCII Code Table

0	<NUL>	32	<SPC>	64	@	96	`	128	Ä	160	†	192	¿	224	‡
1	<SOH>	33	!	65	A	97	a	129	Å	161	°	193	¡	225	·
2	<STX>	34	"	66	B	98	b	130	Ç	162	¢	194	¬	226	‚
3	<ETX>	35	#	67	C	99	c	131	É	163	£	195	√	227	„
4	<EOT>	36	$	68	D	100	d	132	Ñ	164	§	196	ƒ	228	‰
5	<ENQ>	37	%	69	E	101	e	133	Ö	165	•	197	≈	229	Â
6	<ACK>	38	&	70	F	102	f	134	Ü	166	¶	198	∆	230	Ê
7	<BEL>	39	'	71	G	103	g	135	á	167	ß	199	«	231	Á
8	<BS>	40	(	72	H	104	h	136	à	168	®	200	»	232	Ë
9	<TAB>	41	)	73	I	105	i	137	â	169	©	201	…	233	È
10	<LF>	42	*	74	J	106	j	138	ä	170	™	202		234	Í
11	<VT>	43	+	75	K	107	k	139	ã	171	´	203	À	235	Î
12	<FF>	44	,	76	L	108	l	140	å	172	¨	204	Ã	236	Ï
13	<CR>	45	-	77	M	109	m	141	ç	173	≠	205	Õ	237	Ì
14	<SO>	46	.	78	N	110	n	142	é	174	Æ	206	Œ	238	Ó
15	<SI>	47	/	79	O	111	o	143	è	175	Ø	207	œ	239	Ô
16	<DLE>	48	0	80	P	112	p	144	ê	176	∞	208	–	240	
17	<DC1>	49	1	81	Q	113	q	145	ë	177	±	209	—	241	Ò
18	<DC2>	50	2	82	R	114	r	146	í	178	≤	210	“	242	Ú
19	<DC3>	51	3	83	S	115	s	147	ì	179	≥	211	”	243	Û
20	<DC4>	52	4	84	T	116	t	148	î	180	¥	212	‘	244	Ù
21	<NAK>	53	5	85	U	117	u	149	ï	181	µ	213	’	245	ı
22	<SYN	54	6	86	V	118	v	150	ñ	182	∂	214	÷	246	ˆ
23	<ETB>	55	7	87	W	119	w	151	ó	183	∑	215	◊	247	˜
24	<CAN>	56	8	88	X	120	x	152	ò	184	∏	216	ÿ	248	¯
25	<EM>	57	9	89	Y	121	y	153	ô	185	π	217	Ÿ	249	˘
26	<SUB>	58	:	90	Z	122	z	154	ö	186	∫	218	⁄	250	˙
27	<ESC>	59	;	91	[	123	{	155	õ	187	ª	219	€	251	˚
28	<FS>	60	<	92	\	124	|	156	ú	188	º	220	‹	252	¸
29	<GS>	61	=	93	]	125	}	157	ù	189	Ω	221	›	253	˝
30	<RS>	62	>	94	^	126	~	158	û	190	æ	222	ﬁ	254	˛
31	<US>	63	?	95	_	127	<DEL>	159	ü	191	ø	223	ﬂ	255	ˇ