CSE1001 Problem Solving and Programming

L-T-P-J-C

Pre-Req Nil

0- 0- 6-0-3

S.No	Module	Topics	L Hrs	SLO
1	Input and Type Conversion	Newton's Second Law of motion is expressed in the formula $F = m \times a$ where F is force, m is mass, and a is acceleration. Assume that the user knows the mass of an object and the force on that object but wants to obtain the object's acceleration a. Write a program that prompts the user to enter the mass in kilograms (kg) and the force in Newtons (N). The user should enter both values on the same line separated by a comma. Calculate the acceleration using the above formula and display the result to the user. The following demonstrates the proper behavior of the program: 1.Enter the mass in kg and the force in N: 55.4, 6.094 2.The acceleration is 0.11000000000000001	4	1
2	Input and Type Conversion	Write a program which will find all such numbers which are divisible by 7 but are not a multiple of 5, between 2000 and 3200 (both included). The numbers obtained should be printed in a comma separated sequence on a single line	2	1
3	**Functions**	Write a function called calc weight on planet() that calculates your equivalent weight on another planet. This function takes two arguments: your weight on Earth in pounds and the surface gravity of the planet of interest with units m/s2. Make the second argument optional and supply a default value of 23.1 m/s2 which is the approximate surface gravity of Jupiter (Earth's suface gravity is approximately 9.8 m/s2). To perform the conversion, use the equation: weight is equal to mass times suface gravity. Since your weight on Earth is given and you know the Earth's surface gravity, have your function use this information to calculate your mass (it is fine if, at this point, the units of mass are a mix of Imperial and the MKS system). Then, use your mass and the given surface gravity to calculate your effective weight on the other planet.	6	1

		The following demonstrates the proper behavior of this function:
		>>> **calc_weight_on_planet(120, 9.8)**
		120.0
		>>> **calc_weight_on_planet(120)**
		282.85714285714283
		>>> **calc_weight_on_planet(120, 23.1)**
		282.85714285714283		

4	**Functions**	Write a function called num atoms() that calculates how many atoms are in *n* grams of an element given its atomic weight. This function should take two parameters: the amount of the element in grams and an optional argument representing the atomic weight of the element. The atomic weight of any particular element can be found on a periodic table but make the default value for the optional argument the atomic weight of gold (Au) 196.97 with units in grams/mole. A mole is a unit of measurement that is commonly used in chemistry to express an amount of a substance. Avogadro's number is a constant, 6.022 × 1023 atoms/mole, that can be used to find the number of atoms in a given sample. Use Avogadro's number, the atomic weight, and the amount of the element in grams to find the number of atoms present in the sample. Your function should return this value.
		
		The following demonstrates the proper behavior of this function using 10 grams and the
		atomic weight of gold (default), carbon, and hydrogen:
		>>> **num_atoms(10)**
		3.0573183733563486e+22
		>>> **num_atoms(10, 12.001)**
		5.017915173735522e+23
		>>> **num_atoms(10, 1.008)**
		5.97420634920635e+24
			6	1

5	**Recursion**	Write a recursive function to compute the Fibonacci sequence. How does the performance of the recursive function compare to that of an iterative version?	6	14
6	**Strings**	Write a Python program that prompts the user to enter a list of words and stores in a list only those words whose first letter occurs again within the word (for example, 'Baboon'). The program should display the resulting list.	6	14
7	**Strings**	Write a version of a palindrome recognizer that also accepts phrase palindromes such as "Go hang a salami I'm a lasagna hog.", "Was it a rat I saw?", "Step on no pets", "Sit on a potato pan, Otis", "Lisa Bonet ate no basil", "Satan, oscillate my metallic sonatas", "I roamed under it as a tired nude Maori", "Rise to vote sir", or the exclamation "Dammit, I'm mad!". Note that	6	14

		punctuation, capitalization, and spacing are usually ignored.		
8	**Conditional Statements**	In English, the *present participle* is formed by adding the suffix -*ing* to the infinite form: *go -> going*. A simple set of heuristic rules can be given as follows: 1. If the verb ends in *e*, drop the *e* and add *ing* (if not exception: *be, see, flee, knee,* etc.) 2. If the verb ends in *ie*, change *ie* to *y* and add *ing* 3. For words consisting of consonant-vowel-consonant, double the final letter before adding *ing* 4. By default just add *ing* Your task in this exercise is to define a function make_ing_form() which given a verb in infinitive form returns its present participle form. Test your function with words such as *lie*, *see*, *move* and *hug*.	6	14
9	**Loops**	. Define a *procedure* histogram() that takes a list of integers and prints a histogram to the screen. For example, histogram([4, 9, 7]) should print the following: **** ********* *******	4	14
10	**Loops**	A *pangram* is a sentence that contains all the letters of the English alphabet at least once, for example: *The quick brown fox jumps over the lazy dog*. Your task here is to write a function to check a sentence to see if it is a pangram or not.	2	14
11	**Loops**	Write a program to solve a classic ancient Chinese puzzle: We count 35 heads and 94 legs among the chickens and rabbits in a farm. How many rabbits and how many chickens do we have?	2	7
12	**List**	A website requires the users to input username and password to register. Write a program to check the validity of password input by users. Following are the criteria for checking the password:	6	5

		1. At least 1 letter between [a-z] 2. At least 1 number between [0-9] 3. At least 1 character from [$#@] 4. Minimum length of transaction password: 6 5. Maximum length of transaction password: 12 6. At least 1 letter between [A-Z] Your program should accept a sequence of comma separated passwords and will check them according to the above criteria. Passwords that match the criteria are to be printed, each separated by a comma. Example If the following passwords are given as input to the program: ABd1234@1,a F1#,2w3E*,2We3345 Then, the output of the program should be: ABd1234@1		
13	**List**	Write a program that maps a list of words into a list of integers representing the lengths of the correponding words. Write it in three different ways: 1) using a for-loop, 2) using the higher order function map(), and 3) using list comprehensions	4	7
14	**Sorting**	Write a program that prompts the user to enter types of fruit, and how many pounds of fruit there are for each type. The program should then display the information in the form *fruit*, *weight* listed in alphabetical order, one fruit type per line as shown below Apple, 6 lbs.	6	7

		Banana, 11 lbs. etc.		
15	**Sorting**	You are required to write a program to sort the (name, age, height) tuples by ascending order where name is string, age and height are numbers. The tuples are input by console. The sort criteria is: 1: Sort based on name; 2: Then sort based on age; 3: Then sort by score. The priority is that name > age > score. If the following tuples are given as input to the program: Tom,19,80 John,20,90 Jony,17,91 Jony,17,93 Json,21,85 Then, the output of the program should be: [('John', '20', '90'), ('Jony', '17', '91'), ('Jony', '17', '93'), ('Json', '21', '85'), ('Tom', '19', '80')]	6	7
16	**Searching**	In the word game Mad Libs, people are asked to provide a part of speech, such as a noun, verb, adverb, or adjective. The supplied words are used to fill in the blanks of a preexisting template or replace the same parts of speech in a preexisting sentence. Although we don't yet have the tools to implement a full Mad Libs game, we can implement code that demonstrates how the game works for a single sentence. Consider this sentence from P. G. Wodehouse: Jeeves lugged my purple socks out of the drawer as if he were a vegetarian fishing a caterpillar out of his salad. Write a program that will do the following: • Print the following template: Jeeves [verb] my [adjective] [noun] out of the [noun] as if he were a vegetarian fishing a [noun] out of his salad. • Prompt the user for a verb, an adjective, and three nouns. • Print the template with the terms in brackets replaced with the words the user provided. Use string concatenation (i.e., the combining of strings with the	6	2,5

		plus sign) as appropriate. The following demonstrates the proper behavior of this code Jeeves [verb] my [adjective] [noun] out of the [noun] as if he were a vegetarian fishing a [noun] out of his salad. Enter a verb: **bounced** Enter an adjective: **invisible** Enter a noun: **parka** Enter a noun: **watermelon** Enter a noun: **lion** Jeeves bounced my invisible parka out of the watermelon as if he were a vegetarian fishing a lion out of his salad.		
17	**Searching**	In cryptography, a *Caesar cipher* is a very simple encryption techniques in which each letter in the plain text is replaced by a letter some fixed number of positions down the alphabet. For example, with a shift of 3, A would be replaced by D, B would become E, and so on. The method is named after Julius Caesar, who used it to communicate with his generals. *ROT-13* ("rotate by 13 places") is a widely used example of a Caesar cipher where the shift is 13. In Python, the key for ROT-13 may be represented by means of the following dictionary: key = {'a':'n', 'b':'o', 'c':'p', 'd':'q', 'e':'r', 'f':'s', 'g':'t', 'h':'u', 'i':'v', 'j':'w', 'k':'x', 'l':'y', 'm':'z', 'n':'a', 'o':'b', 'p':'c', 'q':'d', 'r':'e', 's':'f', 't':'g', 'u':'h', 'v':'i', 'w':'j', 'x':'k', 'y':'l', 'z':'m', 'A':'N', 'B':'O', 'C':'P', 'D':'Q', 'E':'R', 'F':'S', 'G':'T', 'H':'U', 'I':'V', 'J':'W', 'K':'X', 'L':'Y', 'M':'Z', 'N':'A', 'O':'B', 'P':'C', 'Q':'D', 'R':'E', 'S':'F', 'T':'G', 'U':'H', 'V':'I', 'W':'J', 'X':'K', 'Y':'L', 'Z':'M'} Your task in this exercise is to implement an encoder/decoder of ROT-13. Once you're done, you will be able to read the following secret message: Pnrfne pvcure? V zhpu cersre Pnrfne fnynq! Note that since English has 26 characters, your ROT-13 program will be able to both encode and decode texts written in English	6	2,5
18	**Searching**	In HTML, tags exist in both opening and closing forms and must be balanced to properly describe a web document. This very simple HTML document: <html> <head> <title>	6	2,5

		Example </title> </head> <body> Hello, world </body> </html> is intended only to show the matching and nesting structure for tags in the language. Write a program that can check an HTML document for proper opening and closing tags		
		Total Lab Hours	90	

Text Books

1) Prelude to Programming, 6/E, Stewart Venit Elizabeth Drake, Santa Fe College ISBN-10: 013374163X • ISBN-13: 9780133741636 2015

2) Problem Solving and Programming Concepts, 9/E Maureen Sprankle Jim Hubbard ISBN-10: 0132492644 • ISBN-13: 9780132492645 2012

3) Think like a programmer: An introduction to creative problem solving, Anton Spraul, 2012, No Starch Press, ISBN-10: 1593274246

Reference Books

1) **How to Solve It: A New Aspect of Mathematical Method, G. Polya, Princeton University Press, 2014, ISBN-13: 9780691164076**
2) **How to Solve it By Computer, R.G.Dromey, Pearson Education, 2014, ISBN-13:** 9788131705629
3) **Introduction to Algorithms, 3rd Edition, T.H.Cormen, C.E.Leiserson, R.L.Rivest and C.Stein,MIT Press, 2009, ISBN-13: 9780262533058,**