PROGRAMMING AND COMPUTATIONAL THINKING-XII

(30 MARKS)

Q.NO.	QUESTION & ANSWER	MARKS
1	Explain implicit type conversion and explicit type conversion in Python, with suitable examples?	2
Ans:	In **implicit type conversion**, one data type is automatically converted to another data type. This process doesn't need any user involvement. Ex: X=5 Y=5.5 Z=X+Y print(type(X)) print(type(Y)) print("Z=",Z) print(type(Z)) Output : <class 'int'> <class 'float'> Z=10.5 <class 'float'> Z has float data type because Python always converts smaller data type to larger data type to avoid the loss of data. In **Explicit type conversion**, users convert the data type of an object to required data type. Ex: Mynum=int(input("Enter a number:")) Here, input() returns a string and int() converts it to an integer.	
2	Identify and write the name of the module to which the following functions belong: (i) degrees() (ii) time()	1
Ans.	(i) Math module (ii) datetime module	
3	Observe the following Python code very carefully and rewrite it after removing all syntactical errors with each correction explained. Line 1: def greet(name,msg)	2

	Line 2: print(“Hello”,name,msg) Line 3: greet(“Asha”, “Good Morning”) Line 4: greet(“Good Morning”, name=”John”) Line 5: greet(name=”Pankaj”, “Good Morning”)	
Ans.	**Errors**: Line 1: Colon(:) was missing Line 2: Indentation error Line 4: greet() got multiple values for argument “name” Line 5: Positional argument follows keyword argument **Corrected Code:** def greet(name,msg): print(“Hello”,name,msg) greet(“Asha”, “Good Morning”) greet(“Good Morning”, “John”) greet(name=”Pankaj”, msg=”Good Morning”)	
4	Find the output of the following: word=”Hello Python” print(word.find(‘t’,1)) print(word.find(‘Py’,2)) print(word.find(‘py’,1)) print(word.find(‘o’,5,11))	2
Ans.	8 6 -1 10	
5	Write the output of the following Python program code: def AddList(n,m): L=[] L1=[] L2=[] L1=[i*1 for i in range(n) if i%2==0] L2=[i*2 for i in range(m) if i%3==0]	3

	L.append(L1) L.append(L2) print(L) AddList(5,10)	
Ans.	[[0,2,4], [0,6,12,18]]	
6	Write a recursive function that computes the sum of numbers 1…..n i.e. 1+2+3+…..+n where n is an input from user.	2

Ans.	def sumrec(num):
	if (num==1):
	return 1
	else:
	return(num + sumrec(num-1))
	
	n=int(input("Enter value:"))
	print("Sum of series=", sumrec(n))	

7	How are docstrings different from comments?	2
Ans.	Comments are ignored by the Python Interpreter. Docstrings (though appears like a multiline comment) are executable statements. It can be accessed if placed immediately after a function or class definition or on top of a module. Ex. print(funcevenodd.__doc__)	
8	Write code to print just the last line of a text file 'story.txt'. Also display the total number of lines.	2
Ans.	fin=open("story.txt","r") L=fin.readlines() print("Last Line=" L[-1]) print("Total number of lines=", len(L)) fin.close()	
9	Write a Python program to write a list content to a file.	3
Ans.	color=['Red', 'Green', 'White', 'Black', 'Pink', 'Yellow'] with open('abc.txt',"w") as myfile: for c in color: myfile.write("%s\n",%c) myfile.close()	

	content=open('abc.txt', "r") print(content.read()) content.close()	
10	In what situations would you prefer linear searching over binary searching in terms of effective algorithm?	2
Ans.	Linear Search is preferred over binary search only when:- • The data is not sorted. • The data set is so small that the overhead of a binary search isn't worth the effort.	
11	Write two applications of stack.	2
Ans.	(i) Parenthesis Checking: Stack is used to check the proper opening and closing of parenthesis. (ii) String Reversal: It can be used to reverse a string.	
12	Write Enqueue(), Dequeue() and isEmpty() functions that inserts, deletes and checks for the empty queue, respectively.	3

Ans.	def isEmpty(Qu):
	  if Qu==[]:
	  return True
	  else:
	  return False
	
	def Enqueue(Qu, item):
	  Qu.append(item)
	  if len(Qu)==1:
	  front=rear=0
	  else:
	  rear=len(Qu)-1
	
	def Dequeue(Qu):
	  if isEmpty(Qu):
	  return "Underflow"
	  else:
	  item=Qu.Pop(0)
	  if len(Qu)==0:	

	front=rear=None return item	
13	Create multiple line chart.s on common plot where 4 data ranges are plotted on same chart. The data ranges to be plotted are: Data=[[5., 15., 25., 35.], [9., 18., 21., 15.,], [2., 18., 10., 30.], [13., 27., 20., 35.]]	4
Ans.	import numpy as np import matplotlib.pyplot as plt Data=[[5., 15., 25., 35.], [9., 18., 21., 15.,], [2., 18., 10., 30.], [13., 27., 20., 35.]] x=np.arange(4) plt.plot(x, Data[0], color='b', label='Range1') plt.plot(x, Data[1], color='g', label='Range1') plt.plot(x, Data[2], color='r', label='Range1') plt.plot(x, Data[3], color='y', label='Range1') plt.legend(loc='upper left') plt.title("Multirange Line Chart") plt.xlabel('X') plt.ylabel('Y') plt.show()	