Inspiring Excellence

Course Title: Programming Language I

Course Code: CSE 110

Assignment no: 4

Class Tasks	4
Evaluation Tasks	2 (5+5 Points)
Home Tasks	7 (70 Points)
Total Tasks	12

Class Task 1 [Easy]

1.1) Write a Python program that takes a String as an input from the user and prints that String in **reverse** order.

1.1.1) without using **built-in reverse( ) function** or **slicing** for this task.

1.1.2) using **slicing**.

Sample Input	Output
CSE110	011ESC
Python	nohtyP
1576527	7256751

1.2) Write a python program that takes 2 inputs from the user, where the first input is a string with length greater than 1. The second input is the index of the first given string from where you have to start reversing. After reversing the first input string from that index, print the new string back to the user. See samples below for clarification.

Sample Input	Output
72418 4	81427 **Explanation:** Our second input, index '4' is the last index of our first input String '72418', hence the entire string is reversed giving us '81427'.
12345 2	32145 **Explanation:** The second input is '2' so we have to reverse from index 2 of our first input. The 2nd index of our first input String is '3', index 1 is '2' and index 0 is '1'. Hence, if we reverse indexes 0 to 2, we get '321'. Index 3 and 4 which are '4' and '5' respectively remain unchanged hence our final output is '32145'.
aBcd1234defg 5	21dcBa34defg **Explanation:** Since our second input is 5, index 0 to index 5 is reversed and we have '21dcBa' and the rest is unchanged from indexes 6 to 11 ('34defg'). Therefore, we have '21dcBa34defg' finally.

Class Task 2 [Medium]

2.1) Write a Python program that will ask the user to input a string (containing exactly one word). Then print the ASCII code for each character in the String using the **ord( )** function.

To check if your program is working correctly or not, you can find a list of all correct values from the following website. You may look at "Dec" and "Char" columns only and ignore the other columns for this problem.

ASCII TABLE: http://www.asciitable.com

Sample Input	Output
quincuncial	q: 113 u: 117 i: 105 n: 110 c: 99 u: 117 n: 110 c: 99 i: 105 a: 97 l: 108 **Explanation:** Each line prints a letter sequentially from the given string and its corresponding ASCII value separated by " : ".

2.2) Write a Python program that takes a String as input from the user, removes the characters at even index and prints the resulting String in uppercase.

You cannot use the built-in upper( ) function for this task.

Sample Input	Output
String	TIG **Explanation:** The characters 'S', 'r' and 'n' are at index positions 0, 2, and 4 respectively. Hence they are removed and the remaining characters 'tig' are capitalized giving us output 'TIG'.
abcd	BD **Explanation:** Only the characters at the odd indices, 1 and 3, 'b' and 'd' are capitalized, concatenated and printed.

Class Task 3 [Hard]

Write a python program that splits a given string on a given split character. The first input is a String and the second input is the character that will be used to split the first String.

You cannot use the built-in split() function for this task

Sample Input	Output
This-is-CSE110 -	This is CSE110 **Explanation:** The second input which is the character '-', is used to split or divide the first input String 'This-is-CSE110' into 'This', 'is' and 'CSE110' which are printed individually in separate lines.
tom@gmail,harry@yahoo, bob@gmail,mary@gmail ,	tom@gmail harry@yahoo bob@gmail mary@gmail **Explanation:** The second input which is the character ',' is used to split the first input string 'tom@gmail,harry@yahoo, bol@gmail,mary@gmail' into four separate email addresses.

Class Task 4

```
1  test = ""
2  i = 1
3  j = 0
4  k = 7
5  while (i< 5):
6      k-=1
7      j = k
8      while (j > 4 ):
9          if j % 2 == 0:
10             test = "<--"
11             test = test + str(i) + '3' + "-->" + str(j // 3)
12         else:
13             test = "-->"
14             test = "-->" + str((i // 3)) + test + str(j)
15         print(test)
16         j -=1
17     i+=2
```

Output

Class Evaluation Task 1

Write a program that takes a string as input and prints **"Binary Number"** if the string contains only 0s or 1s. Otherwise, print **"Not a Binary Number"**.

Sample Input	Output
01101101101	Binary Number
12344ab0	Not a Binary Number
10127490111	Not a Binary Number
ary NumberBin	Not a Binary Number

Class Evaluation Task 2

Captain Jack and his crew have discovered a treasure chest full of gold coins. However, the chest comes with a mysterious lock. To open it, they need to input a phrase that should contain a **combination of characters where vowel count is divisible by 3 and consonant count is divisible by 5.** Write a Python program to help Captain Jack determine if the input phrase has the correct number of vowels and consonants to unlock the treasure chest.

Note: Vowels and Consonants count has to be greater than 0 for the treasure to open

Sample Input	Output
Yo-hoo-hoo!	Blimey! No Plunder!! **Explanation:** The input string has five vowels 'o, o, o, o, o' which is not divisible by 3 and the same applies for consonant rules as well where count of consonants is not divisible by 5. Hence the chest cannot be opened.
Yo-ho-Ya-ho-hoo!	Aaarr! Me Plunder!! **Explanation:** The input string has six vowel count which is divisible by 3 and the same applies for

	consonant rules as well where count of consonants is divisible by 5. Hence the chest can be opened.
aoouii-uii	Blimey! No Plunder!! **Explanation:** Here the vowel count is divisible by 3, but the consonant count being 0 resulted in the chest not being opened.

Home Task 1

Given a string, create a new string with all the consecutive duplicates removed.

Hint: You may make a new string to store the result. You can check whether the current character and the next character are the same, then add that character to the new string.

Note: Only consecutive letters are removed, not all duplicate occurrences of a letter. You may try doing this alternative i.e., removing all duplicate letters from a given string, for your own practice.

Sample Input	Output
AAABBBBCDDBBECE	ABCDBECE **Explanation:** From the 3 consecutive "A"s, 2 are removed and we have 'A' only. Then from the 4 consecutives 'B's, 3 are removed and only one is added to the new string giving us "AB". Since we have only one 'C' next, it is added making the resulting string "ABC" now and so on.
Jupyter Notebook is better. Case sensitivity check AAaaaAaaAAAa.	Jupyter Notebok is beter. Case sensitivity check AaAaAa. **Explanation:** Just the 2 consecutive 'o's and 't's are changed to one at first and the uppercase 'A' and lowercase 'a' are treated separately i.e., case sensitive when checking for consecutive duplicates.

Home Task 2

Write a Python program that will take one input from the user made up of two strings separated by a comma and a space (see samples below). Then create a mixed string with alternative characters from each string. Any leftover chars will be appended at the end of the resulting string. **[Do not use lists for this task]**

Hint: For adding the leftover characters you may use string slicing.

Sample Input	Output
ABCD, efgh	AeBfCgDh **Explanation:** At first, the two strings divided by ", " should be separated. Then the first character of the first string 'A' is concatenated with the first character of the second string 'e' which will concatenate to the second character of the first string 'B', the second character of the second-string f and so on since the strings are of equal length.

ABCDENDFGH, ijkl	AiBjCkDlENDFGH **Explanation:** Here, since the length of the first string is greater than the length of the second string, after separation, the characters are concatenated alternatively as in sample input/output 1, till the length of the second string i.e., ijkl. Since, there are no more characters in the second string after that, the remaining characters of the first string i.e., ENDFGH in this case are concatenated at the end of the final string.
ijkl, ABCDENDFGH	iAjBkClDENDFGH **Explanation**: This time, the length of the second string is greater than the length of the first string therefore the first letters of the 2 strings 'i' and 'A', then the second letters 'j' and 'B' and so on are being concatenated until there are no more letters in the first shorter string following which the remaining letters i.e., ENDFGH again in this case too (this may be different for other different string inputs) are added at the end giving us the resulting output string.

Home Task 3

Write a Python program that takes a string as an input from the user containing all small letters and then prints the next alphabet in sequence for each alphabet in the input.

Hint: You may use the functions **ord( )** and **chr( )**. The ASCII value of ‘a’ is 97 and ‘z’ is 122.

Sample Input	Output
abcd	bcde
the cow	uif!dpx
xyzabc	yzabcd

Home Task 4

Write a python program that takes a string as an input from the user and then modifies the string in such a way that the string always starts with an uppercase letter and the case of each subsequent letter is the opposite of the previous letter (uppercase character followed by a lowercase character followed by an uppercase character and so on). Finally, the modified string is printed to show the user.

Hint: Flags/counters can be used to manage uppercase-lowercase

Sample Input	Output
Python programming is very easy	PyThOn PrOgRaMmInG iS vErY eAsY
I love Python Programming	I lOvE pYtHoN pRoGrAmMiNg
CSE110 Course	CsE110 cOuRsE
c	C

Home Task 5

```
1  i = 10
2  while(i >= -20):
3      if(i < 0):
4          test = " != "
5          test = str(i//2) + test + str(int(i/2))
6      else:
7          test = " == "
8          test = str(i//2) + test + str(int(i/2))
9      print(test)
10     i -= 5
```

Create an output table/column to show the final answer for the above tracing.

Home Task 6

```
1  test = ""
2  i = 5
3  j = 0
4  k = 15
5  while i< 10:
6      k-=1
7      j = k
8      while 5 < j - 5:
9          if (j % 2) == 0:
10             test = "<--"
11             test = str(test) + str(i) + str(2) + "-->" + str(int(j / 2))
12         else:
13             test = "-->"
14             test = "-->" + str(int(i / 2)) + str(test) + str(j)
15         print(test)
16         j = j-1
17     i+=1
```

Create a proper trace table and an output column to show the final answer for the above tracing.

Home Task 7

```
1  i=0
2  j=0
3  k=15
4  test = '<--cat'
5  while i < 5:
6      k -= 1
7      j = k
8      while j > 10:
9          if j % 2 == 0:
10             test += '-->'
11             test = test + str(i) + str(j // 2)
12         else:
13             test += '<--'
14             test = test + str(i // 2) + str(j)
15         print(test)
16         j-=1
17     i+=1
```

Create a proper trace table and an output column to show the final answer for the above tracing.