# Rajalakshmi Engineering College

Name: Neha s
Email: 240701355@rajalakshmi.edu.in
Roll no: 240701355
Phone: 9884059069
Branch: REC
Department: I CSE FD
Batch: 2028
Degree: B.E - CSE

Scan to verify results

## NeoColab_REC_CS23221_Python Programming

### REC_Python_Week 6_MCQ

Attempt : 1
Total Mark : 20
Marks Obtained : 20

### Section 1 : MCQ

1. Fill in the blanks in the following code of writing data in binary files.

```
import _____ (1)
rec=[]
while True:
    rn=int(input("Enter"))
    nm=input("Enter")
    temp=[rn, nm]
    rec.append(temp)
    ch=input("Enter choice (y/N)")
    if ch.upper=="N":
        break
f.open("stud.dat","_____")(2)
_____.dump(rec,f)(3)
_____.close()(4)
```

***Answer***

(pickle,wb,pickle,f)

***Status :*** Correct ***Marks : 1/1***

2. Match the following:

a) f.seek(5,1) i) Move file pointer five characters behind from the current position

b) f.seek(-5,1) ii) Move file pointer to the end of a file

c) f.seek(0,2) iii) Move file pointer five characters ahead from the current position

d) f.seek(0) iv) Move file pointer to the beginning of a file

***Answer***

a-iii, b-i, c-ii, d-iv

***Status :*** Correct ***Marks : 1/1***

3. What is the default value of reference_point in the following code?

file_object.seek(offset [,reference_point])

***Answer***

0

***Status :*** Correct ***Marks : 1/1***

4. What is the correct way to raise an exception in Python?

***Answer***

raise Exception()

***Status :*** Correct ***Marks : 1/1***

5. What will be the output of the following Python code?

```
# Predefined lines to simulate the file content
lines = [
    "This is 1st line",
    "This is 2nd line",
    "This is 3rd line",
    "This is 4th line",
    "This is 5th line"
]

print("Name of the file: foo.txt")

# Print the first 5 lines from the predefined list
for index in range(5):
    line = lines[index]
    print("Line No %d - %s" % (index + 1, line.strip()))
```

***Answer***

Displays Output

***Status :*** Correct ***Marks : 1/1***

6. Which of the following is true about

fp.seek(10,1)

***Answer***

Move file pointer ten characters ahead from the current position

***Status :*** Correct ***Marks : 1/1***

7. How do you rename a file?

***Answer***

os.rename(existing_name, new_name)

***Status :*** Correct ***Marks : 1/1***

8. What is the purpose of the except clause in Python?

***Answer***

To handle exceptions during code execution

***Status :*** Correct ***Marks : 1/1***

9. What is the output of the following code?

```
class MyError(Exception):
    pass

try:
    raise MyError("Something went wrong")
except MyError as e:
    print(e)
```

***Answer***

Something went wrong

***Status :*** Correct ***Marks : 1/1***

10. What is the output of the following code?

```
try:
    x = "hello" + 5
except TypeError:
    print("Type Error occurred")
finally:
    print("This will always execute")
```

***Answer***

Type Error occurredThis will always execute

***Status :*** Correct ***Marks : 1/1***

11. What is the output of the following code?

```
try:
    x = 1 / 0
except ZeroDivisionError:
    print("Caught division by zero error")
finally:
    print("Executed")
```

***Answer***

Caught division by zero errorExecuted

***Status :*** Correct ***Marks : 1/1***

12. Fill the code to in order to read file from the current position.

Assuming exp.txt file has following 3 lines, consider current file position is beginning of 2nd line

Meri,25

John,21

Raj,20

Ouptput:

['John,21\n','Raj,20\n']

```
f = open("exp.txt", "w+")
_____(1)
print _____(2)
```

***Answer***

1) f.seek(0, 1)2) f.readlines()

***Status :*** Correct ***Marks : 1/1***

13. Which clause is used to clean up resources, such as closing files in Python?

***Answer***

finally

***Status :*** Correct **Marks : 1/1**

14. Fill in the code in order to get the following output:

Output:

Name of the file: ex.txt

```
fo = open(_____(1), "wb")
print("Name of the file: ",_____)(2)
```

***Answer***

1) "ex.txt"2) fo.name

***Status :*** Correct **Marks : 1/1**

15. How do you create a user-defined exception in Python?

***Answer***

By creating a new class that inherits from the Exception class

***Status :*** Correct **Marks : 1/1**

16. What happens if no arguments are passed to the seek function?

***Answer***

file position remains unchanged

***Status :*** Correct **Marks : 1/1**

17. What will be the output of the following Python code?

```
f = None
for i in range (5):
   with open("data.txt", "w") as f:
      if i > 2:
         break
```

print(f.closed)

***Answer***

True

***Status :*** Correct ***Marks : 1/1***

18. Which of the following is true about the finally block in Python?

***Answer***

The finally block is always executed, regardless of whether an exception occurs or not

***Status :*** Correct ***Marks : 1/1***

19. What happens if an exception is not caught in the except clause?

***Answer***

The program will display a traceback error and stop execution

***Status :*** Correct ***Marks : 1/1***

20. What is the difference between r+ and w+ modes?

***Answer***

in r+ the pointer is initially placed at the beginning of the file and the pointer is at the end for w+

***Status :*** Correct ***Marks : 1/1***

# Rajalakshmi Engineering College

Name: Neha s
Email: 240701355@rajalakshmi.edu.in
Roll no: 240701355
Phone: 9884059069
Branch: REC
Department: I CSE FD
Batch: 2028
Degree: B.E - CSE

Scan to verify results

## NeoColab_REC_CS23221_Python Programming

## REC_Python_Week 6_COD

Attempt : 1
Total Mark : 50
Marks Obtained : 50

## Section 1 : Coding

1. Problem Statement

A retail store requires a program to calculate the total cost of purchasing a product based on its price and quantity. The program performs validation to ensure valid inputs and handles specific error conditions using exceptions:

Price Validation: If the price is zero or less, raise a ValueError with the message: "Invalid Price".Quantity Validation: If the quantity is zero or less, raise a ValueError with the message: "Invalid Quantity".Cost Threshold: If the total cost exceeds 1000, raise RuntimeError with the message: "Excessive Cost".

***Input Format***

The first line of input consists of a double value, representing the price of a product.

The second line consists of an integer, representing the quantity of the product.

***Output Format***

If the calculation is successful, print the total cost rounded to one decimal place.

If the price is zero or less prints "Invalid Price".

If the quantity is zero or less prints "Invalid Quantity".

If the total cost exceeds 1000, prints "Excessive Cost".

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: 20.0
5
Output: 100.0

***Answer***

```
try:
    price = float(input())
    quantity=int(input())
    if price <= 0:
        raise ValueError("Invalid Price")
    if quantity <= 0:
        raise ValueError("Invalid Quantity")
    total_cost=price * quantity
    if total_cost > 1000:
        raise RuntimeError("Excessive cost")
    print(f"{total_cost :.1f}")
except ValueError as e:
    if str(e) in ["Invalid Price", "Invalid Quantity"]:
      print(e)
    else:
        print("Invalid price")
except RuntimeError as e:
    print(e)
```

*Status :* Correct *Marks : 10/10*

2. Problem Statement

Write a program that calculates the average of a list of integers. The program prompts the user to enter the length of the list (n) and each element of the list. It performs error handling to ensure that the length of the list is a non-negative integer and that each input element is a numeric value.

***Input Format***

The first line of the input is an integer n, representing the length of the list as a positive integer.

The second line of the input consists of an element of the list as an integer, separated by a new line.

***Output Format***

If the length of the list is not a positive integer or zero, the output displays "Error: The length of the list must be a non-negative integer."

If a non-numeric value is entered for the length of the list, the output displays "Error: You must enter a numeric value."

If a non-numeric value is entered for a list element, the output displays "Error: You must enter a numeric value."

If the inputs are valid, the program calculates and prints the average of the provided list of integers with two decimal places: "The average is: [average]".

Refer to the sample output for the formatting specifications.

***Sample Test Case***

Input: -2

1
2
Output: Error: The length of the list must be a non-negative integer.

***Answer***

```
try:
    n_input=input()
    n=int(n_input)

    if n<=0:
        print("Error: The length of the list must be a non-negative integer.")
    else:
        numbers=[]
        for i in range(n):
            try:
                element=input()
                number=int(element)
                numbers.append(number)
            except ValueError:
                print("Error: you must enter a numeric value.")
                break
        else:
            avg=sum(numbers)/n
            print(f"The average is: {avg:.2f}")

except ValueError:
    print("Error: you must enter a numeric value.")
```

***Status :*** Correct ***Marks : 10/10***

3. Problem Statement

In a voting system, a person must be at least 18 years old to be eligible to vote. If a user enters an age below 18, the system should raise a user-defined exception indicating that they are not eligible to vote.

***Input Format***

The input contains a positive integer representing age.

***Output Format***

If the age is less than 18, the output displays "Not eligible to vote".

Otherwise, the output displays "Eligible to vote".

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: 18

Output: Eligible to vote

***Answer***

```
class AgeEligibility(Exception):
    pass
try:
    age = int(input())
    if age<18:
        raise AgeEligibility("Not eligible to vote")
    print("Eligible to vote")

except AgeEligibility as e:
    print(e)
except ValuError:
    print("Not eligible to vote");
```

***Status :*** Correct ***Marks : 10/10***

4. Problem Statement

Sophie enjoys playing with words and wants to count the number of words in a sentence. She inputs a sentence, saves it to a file, and then reads it from the file to count the words.

Write a program to determine the number of words in the input sentence.

File Name: sentence_file.txt

***Input Format***

The input consists of a single line of text containing words separated by spaces.

***Output Format***

The output displays the count of words in the sentence.

Refer to the sample output for the formatting specifications.

***Sample Test Case***

Input: Four Words In This Sentence
Output: 5

***Answer***

```
sen = input()
with open('sen_file.txt','w') as file:
    file.write(sen)
with open('sen_file.txt','r') as file:
    read_sen=file.read()

words = read_sen.split()
words_count= len(words)
print(words_count)
```

***Status :*** Correct ***Marks : 10/10***

5. Problem Statement

Tara is a content manager who needs to perform case conversions for various pieces of text and save the results in a structured manner.

She requires a program to take a user's input string, save it in a file, and then retrieve and display the string in both upper-case and lower-case versions. Help her achieve this task efficiently.

File Name: text_file.txt

***Input Format***

The input consists of a single line containing a string provided by the user.

***Output Format***

The first line displays the original string read from the file in the format: "Original String: {original_string}".

The second line displays the upper-case version of the original string in the format: "Upper-Case String: {upper_case_string}".

The third line displays the lower-case version of the original string in the format: "Lower-Case String: {lower_case_string}".

Refer to the sample output for the formatting specifications.

***Sample Test Case***

Input: #SpecialSymBoLs1234

Output: Original String: #SpecialSymBoLs1234
Upper-Case String: #SPECIALSYMBOLS1234
Lower-Case String: #specialsymbols1234

***Answer***

```
input_string=input()

if not(5<=len(input_string) <= 500):
    print("Error")

else:
    file_name="text_file.txt"

    with open(file_name,"w") as file:
        file.write(input_string)
    with open(file_name,"r") as file:
        original_string=file.read()

    upper_case=original_string.upper()
    lower_case=original_string.lower()
```

```
print(f"Original String: {original_string}")
print(f"Upper-case String: {upper_case}")
print(f"Lower-case String: {lower_case}")
```

***Status :*** Correct ***Marks : 10/10***

# Rajalakshmi Engineering College

Name: Neha s
Email: 240701355@rajalakshmi.edu.in
Roll no: 240701355
Phone: 9884059069
Branch: REC
Department: I CSE FD
Batch: 2028
Degree: B.E - CSE

Scan to verify results

## NeoColab_REC_CS23221_Python Programming

## REC_Python_Week 6_CY

Attempt : 1
Total Mark : 40
Marks Obtained : 40

## Section 1 : Coding

1. Problem Statement

Write a program to read the Register Number and Mobile Number of a student. Create user-defined exception and handle the following:

If the Register Number does not contain exactly 9 characters in the specified format(2 numbers followed by 3 characters followed by 4 numbers) or if the Mobile Number does not contain exactly 10 characters, throw an IllegalArgumentException. If the Mobile Number contains any character other than a digit, raise a NumberFormatException.If the Register Number contains any character other than digits and alphabets, throw a NoSuchElementException.If they are valid, print the message ‘valid’ or else print an Invalid message.

***Input Format***

The first line of the input consists of a string representing the Register number.

The second line of the input consists of a string representing the Mobile number.

***Output Format***

The output should display any one of the following messages:

If both numbers are valid, print "Valid".

If an exception is raised, print "Invalid with exception message: ", followed by the specific exception message.

Refer to the sample output for the formatting specifications.

***Sample Test Case***

Input: 19ABC1001
9949596920

Output: Valid

***Answer***

```
class InvalidRegisterNumberException(Exception):
    def __init__(self, message):
        super().__init__(message)


class InvalidMobileNumberException(Exception):
    def __init__(self, message):
        super().__init__(message)


try:
    register_number = input()

    if len(register_number) != 9:
        raise InvalidRegisterNumberException("Register Number should have
exactly 9 characters.")

    if not register_number[:2].isdigit() or not register_number[2:5].isalpha() or not
register_number[5:].isdigit():
        raise InvalidRegisterNumberException("Register Number should have the
```

```
format: 2 numbers, 3 characters, and 4 numbers.")

    mobile_number = input()

    if len(mobile_number) != 10:
        raise InvalidMobileNumberException("Mobile Number should have exactly
10 characters.")

    if not mobile_number.isdigit():
        raise InvalidMobileNumberException("Mobile Number should only contain
digits.")

    print("Valid")

except InvalidRegisterNumberException as e:
    print("Invalid with exception message:", str(e))

except InvalidMobileNumberException as e:
    print("Invalid with exception message:", str(e))

except Exception as e:
    print("Invalid with exception message:", str(e))
```

***Status :*** Correct ***Marks : 10/10***

2. Problem Statement

Bob, a data analyst, requires a program to automate the process of analyzing character frequency in a given text. This program should allow the user to input a string, calculate the frequency of each character within the text, save these character frequencies to a file named "char_frequency.txt," and display the results.

***Input Format***

The input consists of the string.

***Output Format***

The first line prints "Character Frequencies:".

The following lines print the character frequency in the format: "X: Y" where X is

the character and Y is the count.

Refer to the sample output for the formatting specifications.

***Sample Test Case***

Input: aaabbbccc

Output: Character Frequencies:
a: 3
b: 3
c: 3

***Answer***

```
user_string = input()

char_frequency = {}
for char in user_string:
    if char in char_frequency:
        char_frequency[char] += 1
    else:
        char_frequency[char] = 1

with open("char_frequency.txt", "w") as file:
    for char, frequency in char_frequency.items():
        file.write(f"{char}: {frequency}\n")

with open("char_frequency.txt", "r") as file:
    frequencies = file.read()

print("Character Frequencies:")
print(frequencies)
```

***Status :*** Correct ***Marks : 10/10***

3. Problem Statement

In the enchanted realm of Academia, you, the Academic Alchemist, are

bestowed with a magical quill and a parchment to weave the grades of aspiring students into a tapestry of academic brilliance.

The mission is to craft a Python program that empowers faculty members to enter student grades for any two subjects, stores these magical grades in a mystical file, and then, with a wave of your virtual wand, calculates the GPA to unveil the true essence of academic achievement.

***Input Format***

The input format is a string representing the student's name, any two subjects, and corresponding grades.

After entering grades, they can type 'done' when prompted for the student's name.

***Output Format***

The output should display the (average of grades) calculated GPA with a precision of two decimal places.

The magical grades will be saved in a mystical file named "magical_grades.txt".

Refer to the sample output for format specifications.

***Sample Test Case***

Input: Alice
Math
95
English
88
done

Output: 91.50

***Answer***

```
magical_grades = {}

while True:
    student_name = input()
```

```
    if student_name.lower() == 'done':
        break

    subjects_grades = {}

    for _ in range(2):
        subject = input()
        if subject.lower() == 'done':
            break

        grade_input = input()
        grade = float(grade_input) if grade_input.replace('.', '', 1).isdigit() else -1

        while not (0 <= grade <= 100):
            grade_input = input()
            grade = float(grade_input) if grade_input.replace('.', '', 1).isdigit() else -1

        subjects_grades[subject] = grade

    magical_grades[student_name] = subjects_grades

total_gpa = sum(sum(grades.values()) for grades in magical_grades.values())
total_subjects = sum(len(grades) for grades in magical_grades.values())
gpa = total_gpa / total_subjects if total_subjects != 0 else 0

with open("magical_grades.txt", "w") as file:
    for student, grades in magical_grades.items():
        file.write("{}: {}\n".format(student, grades))
print("{:.2f}".format(gpa))
```

***Status :*** Correct **Marks : 10/10**

4. Problem Statement

A shopkeeper is recording the daily sales of an item for N days, where the price of the item remains the same for all days. Write a program to calculate the total sales for each day and save them in a file named sales.txt that can store the data for a maximum of 30 days. Then, read the file and display the total earnings for each day.

Note: Total Earnings for each day = Number of Items sold in that day × Price of the item.

***Input Format***

The first line of input consists of an integer N, representing the number of days.

The second line of input consists of N space-separated integers representing the number of items sold each day.

The third line of input consists of an integer M, representing the price of the item that is common for all N days.

***Output Format***

If the number of days entered exceeds 30 (N > 30), the output prints "Exceeding limit!" and terminates.

Otherwise, the code reads the contents of the file and displays the total earnings for each day on separate lines.

Contents of the file: The total earnings for N days, with each day's earnings appearing on a separate line.

Refer to the sample output for the formatting specifications.

***Sample Test Case***

Input: 4
5 10 5 0
20

Output: 100
200
100
0

***Answer***

```
N = int(input())
```

```
items_input = input().split()
price = int(input())

if N > 30:
    print("Exceeding limit!")
else:
    items = [int(item) for item in items_input]

    with open("sales.txt", "w") as of:
        for item in items:
            of.write(str(item * price) + '\n')

    with open("sales.txt", "r") as inputFile:
        for line in inputFile:
            print(int(line))
```

***Status :*** Correct **Marks : 10/10**

# Rajalakshmi Engineering College

Name: Neha s
Email: 240701355@rajalakshmi.edu.in
Roll no: 240701355
Phone: 9884059069
Branch: REC
Department: I CSE FD
Batch: 2028
Degree: B.E - CSE

Scan to verify results

## NeoColab_REC_CS23221_Python Programming

## REC_Python_Week 6_PAH

Attempt : 1
Total Mark : 30
Marks Obtained : 30

## Section 1 : Coding

1. Problem Statement

John is a data analyst who often works with text files. He needs a program that can analyze the contents of a text file and count the number of times a specific character appears in the file.

John wants a simple program that allows him to specify a file and a character to count within that file.

***Input Format***

The first line of input consists of the file's name to be analyzed.

The second line of the input consists of the string they want to write within the file.

The third line of the input consists of a character to count within the file.

***Output Format***

If the character is found, the output displays "The character 'X' appears {Y} times in the file." where X is the character and Y i the count,

If the character does not appear in the file, the output displays "Character not found."

Refer to the sample output for the formatting specifications.

***Sample Test Case***

Input: test.txt
This is a test file to check the character count.
e

Output: The character 'e' appears 5 times in the file.

***Answer***

```
def count_character_in_file(file_name, character):
    try:
        with open(file_name, 'r') as file:
            content = file.read()
            count = content.lower().count(character.lower())
            return count
    except FileNotFoundError:
        return -1

file_name = input()
content = input()
character_to_count = input()

with open(file_name, 'w') as file:
    file.write(content)

result = count_character_in_file(file_name, character_to_count)

if result == -1:
```

```
    print("File not found.")
elif result == 0:
    print("Character not found in the file.")
else:
    print(f"The character '{character_to_count}' appears {result} times in the file.")
```

***Status :*** Correct ***Marks : 10/10***

2. Problem Statement

Reeta is playing with numbers. Reeta wants to have a file containing a list of numbers, and she needs to find the average of those numbers. Write a program to read the numbers from the file, calculate the average, and display it.

File Name: user_input.txt

***Input Format***

The input file will contain a single line of space-separated numbers (as a string).

These numbers may be integers or decimals.

***Output Format***

If all inputs are valid numbers, the output should print: "Average of the numbers is: X.XX" (where X.XX is the computed average rounded to two decimal places)

If the input contains invalid data, print: "Invalid data in the input."

Refer to the sample output for format specifications.

***Sample Test Case***

Input: 1 2 3 4 5
Output: Average of the numbers is: 3.00

***Answer***

```
def calculate_average(numbers):
```

```
    if not numbers:
        return "No numbers provided."

    if all(num.replace('.', '', 1).isdigit() or num.isdigit() for num in numbers):
        numbers = [float(num) for num in numbers]
        average = sum(numbers) / len(numbers)
        return f"Average of the numbers is: {average:.2f}"
    else:
        return "Invalid data in the input."

user_input = input()

input_numbers = user_input.split()

with open("user_input.txt", "w") as file:
    file.write(','.join(input_numbers))

with open("user_input.txt", "r") as file:
    file_numbers = file.read().split(',')

result = calculate_average(file_numbers)
print(result)
```

***Status :*** Correct **Marks : 10/10**

3. Problem Statement

Peter manages a student database and needs a program to add students. For each student, Alex inputs their ID and name. The program checks for duplicate IDs and ensures the database isn't full.

If a duplicate or a full database is detected, an appropriate error message is displayed. Otherwise, the student is added, and a confirmation message is shown. The database has a maximum capacity of 30 students, and each student must have a unique ID.

***Input Format***

The first line contains an integer n, representing the number of students to be added to the school database.

The next n lines each contain two space-separated values, representing the student's ID (integer) and the student's name (string).

***Output Format***

The output will depend on the actions performed in the code.

If a student is added to the database, the output will display: "Student with ID [ID number] added to the database."

If there is an exception due to a duplicate student ID, the output will display: "Exception caught. Error: Student ID already exists."

If there is an exception due to the database being full, the output will display: "Exception caught. Error: Student database is full."

Refer to the sample outputs for the formatting specifications.

***Sample Test Case***

Input: 3
16 Sam
87 Sabari
43 Dani

Output: Student with ID 16 added to the database.
Student with ID 87 added to the database.
Student with ID 43 added to the database.

***Answer***

```
MAX_CAPACITY = 30
students = []
num_students = 0

def error_message_duplicate_id():
    return "Error: Student ID already exists."

def error_message_full_database():
    return "Error: Student database is full."
```

```
def add_student(student_id, student_name):
    global num_students

    if num_students >= MAX_CAPACITY:
        raise Exception(error_message_full_database())

    for existing_student in students:
        if existing_student['id'] == student_id:
            raise Exception(error_message_duplicate_id())

    student = {'id': student_id, 'name': student_name}
    students.append(student)
    num_students += 1

    print(f"Student with ID {student_id} added to the database.")

if __name__ == "__main__":
    try:
        n = int(input())

        for _ in range(n):
            id_input, name_input = map(str, input().split())
            add_student(int(id_input), name_input)

    except Exception as e:
        if str(e) == error_message_duplicate_id():
            print(f"Exception caught. {error_message_duplicate_id()}")
        elif str(e) == error_message_full_database():
            print(f"Exception caught. {error_message_full_database()}")
```

***Status :*** Correct **Marks : 10/10**