# Rajalakshmi Engineering College

Name: RANNESH KHUMAR B R
Email: 240701422@rajalakshmi.edu.in
Roll no: 2116240701
Phone: 9042350670
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 IllegalArgumentException(Exception):
   pass

class NumberFormatException(Exception):
   pass

class NoSuchElementException(Exception):
   pass

def validate_register_number(reg_num):
   if len(reg_num) != 9:
      raise IllegalArgumentException("Register Number should have exactly 9 
characters.")
   if not (reg_num[:2].isdigit() and reg_num[2:5].isalpha() and 
reg_num[5:].isdigit()):
      raise IllegalArgumentException("Register Number should have the format: 2 
numbers, 3 characters, and 4 numbers.")
   for char in reg_num:
      if not (char.isdigit() or char.isalpha()):
         raise NoSuchElementException("Register Number should only contain
digits and alphabets.")

def validate_mobile_number(mobile_num):
   if len(mobile_num) != 10:
      raise IllegalArgumentException("Mobile Number should have exactly 10
characters.")
   if not mobile_num.isdigit():
      raise NumberFormatException("Mobile Number should only contain digits.")

def main():
   reg_num = input().strip()
   mobile_num = input().strip()
   try:
      validate_register_number(reg_num)
      validate_mobile_number(mobile_num)
      print("Valid")
   except IllegalArgumentException as e:
      print(f"Invalid with exception message: {str(e)}")
   except NumberFormatException as e:
      print(f"Invalid with exception message: {str(e)}")
   except NoSuchElementException as e:
      print(f"Invalid with exception message: {str(e)}")

if __name__ == "__main__":
   main()
```

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

2. Problem Statement

Write a program to obtain the start time and end time for the stage event show. If the user enters a different format other than specified, an exception occurs and the program is interrupted. To avoid that, handle the exception and prompt the user to enter the right format as specified.

Start time and end time should be in the format 'YYYY-MM-DD HH:MM:SS'If the input is in the above format, print the start time and end time.If the input does not follow the above format, print "Event time is not in the format "

***Input Format***

The first line of input consists of the start time of the event.

The second line of the input consists of the end time of the event.

***Output Format***

If the input is in the given format, print the start time and end time.

If the input does not follow the given format, print "Event time is not in the format".

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: 2022-01-12 06:10:00
2022-02-12 10:10:12

Output: 2022-01-12 06:10:00
2022-02-12 10:10:12

***Answer***

```
from datetime import datetime

def validate_time(time_str):
    try:
        datetime.strptime(time_str, '%Y-%m-%d %H:%M:%S')
        return True
    except ValueError:
        return False

start_time = input().strip()
end_time = input().strip()

if validate_time(start_time) and validate_time(end_time):
    print(start_time)
    print(end_time)
else:
    print("Event time is not in the format")
```

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

3. Problem Statement

Alex is creating an account and needs to set up a password. The program prompts Alex to enter their name, mobile number, chosen username, and desired password. Password validation criteria include:

Length between 10 and 20 characters.At least one digit.At least one special character from !@#$%^&amp;* set. Display "Valid Password" if criteria are met; otherwise, raise an exception with an appropriate error message.

***Input Format***

The first line of the input consists of the name as a string.

The second line of the input consists of the mobile number as a string.

The third line of the input consists of the username as a string.

The fourth line of the input consists of the password as a string.

***Output Format***

If the password is valid (meets all the criteria), it will print "Valid Password"

If the password is weak (fails any one or more criteria), it will print an error message accordingly.

Refer to the sample outputs for the formatting specifications.

***Sample Test Case***

Input: John
9874563210
john
john1#nhoj
Output: Valid Password

***Answer***

```
name = input().strip()
mobile = input().strip()
```

```
username = input().strip()
password = input().strip()

special_chars = {'!', '@', '#', '$', '%', '^', '&', '*'}
errors = []


if len(password) < 10 or len(password) > 20:
    errors.append("Should be a minimum of 10 characters and a maximum of 20
characters")
else:
    if not any(c.isdigit() for c in password):
        errors.append("Should contain at least one digit")
    if not any(c in special_chars for c in password):
        errors.append("It should contain at least one special character")

if not errors:
    print("Valid Password")
else:
    for error in errors:
        print(error)
```

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

4. 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***

```
input_string = input().strip()


frequency = {}


for char in input_string:
    if char in frequency:
        frequency[char] += 1
    else:
        frequency[char] = 1


print("Character Frequencies:")
for char, count in (frequency.items()):
    print(f"{char}: {count}")


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

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