CSV File Handling

Question 1

Write definitions for the following user defined functions
a)def Write_info() : To insert a record into a csv file named "Product.csv"
(Ensure that the header row is written only once into a file) with the following details: [prod_id,prod_name, prod_price, prod_qty] with ;(semicolon as delimiter)
b)def Read_info() : To display all the records stored in the csv file passed as the parameter.(Display appropriate messages if file doesn't exist or if there are zero records)
Create a Menu as follows and invoke the required function defined above to execute the task chosen by the user.

Source Code:

```
import csv
def Write_info():
    try:
        with open ('Product.csv','r') as file:
            pass
    except FileNotFoundError:
        with open ('Product.csv','w',newline='') as file:
            writer=csv.writer(file,delimiter=';')
            writer.writerow(['prod_id' ,'prod_name',
'prod_price', 'prod_qty'])
    with open('Product.csv','a',newline='') as a_file:
        writer=csv.writer(a_file,delimiter=';')
        id=input('Enter the Product ID:')
        name=input('Enter the Name:')
        price=int(input('Enter the price:'))
        qty=int(input('Enter the Quantity:'))
        writer.writerow([id,name,price,qty])
def Read_info(filename):
    try:
        with open(filename, mode='r') as file:
```

```
            reader = csv.reader(file, delimiter=';')
            data = list(reader)
            if len(data) > 1:
                for row in data[0:]:
                    print(row)
            else:
                print("No records found.")
    except FileNotFoundError:
        print("File not found.")
while True:
    print('''MENU:
1.Insert a New Record (Retain previous data)
2.View all Records
3.Exit ''')
    ch=int(input("Enter the Choice Number:"))
    if ch==1:
        Write_info()
    elif ch==2:
        Read_info(input("Enter the File Name:"))
    elif ch==3:
        break
    else:
        print('Invalid Option Enter the Correct Option')
```

Output:

```
MENU:
1.      Insert a New Record (Retain previous data)
2.      View all Records
3.      Exit
Enter the Choice Number:1
Enter the Product ID:12
Enter the Name:Hello
Enter the price:123
Enter the Quantity:1
MENU:
1.      Insert a New Record (Retain previous data)
```

```
2.      View all Records
3.      Exit
Enter the Choice Number:2
Enter the File Name:Product.csv
['prod_id', 'prod_name', 'prod_price', 'prod_qty']
['1', 'Hi', '1', '1']
['2', 'Hello', '23', '1']
['12', 'Hello', '123', '1']
MENU:
1.      Insert a New Record (Retain previous data)
2.      View all Records
3.      Exit
Enter the Choice Number:4
Invalid Option Enter the Correct Option
MENU:
1.      Insert a New Record (Retain previous data)
2.      View all Records
3.      Exit
Enter the Choice Number:3
```

Question 2

Write a Python program
a)To write a nested list(with 3 records) with the following details [M_type,M_name,M_price] to a “Medicine.csv” in one go. (M_type takes the value “safe ” and “unsafe”)
b) After writing the content to “Medicine.csv” file read and display the content.

Source Code:

```
import csv
def medicine():
    with open('Medicine.csv', mode='w', newline='') as
file:
        writer = csv.writer(file)
        writer.writerow(['M_type','M_name','M_price'])
        def M_typechk():
            a = input("Enter the Medicine Type Safe/Unsafe:
")
            if a.lower() in ['safe', 'unsafe']:
                return a
            else:
                print('Invalid Medicine Type')
                return M_typechk()
        for l in range(3):
            M_name=input("Enter the Medicine Name:")
            M_price=int(input("Enter the Price"))
            M_type=M_typechk()
            writer.writerow([M_type,M_name,M_price])
    with open('Medicine.csv', mode='r') as file:
        reader = csv.reader(file)
        for row in reader:
            print(row)
medicine()
```

Output:

```
Enter the Medicine Name:Dolo-650
Enter the Price12
Enter the Medicine Type Safe/Unsafe: Safe
Enter the Medicine Name:Dolo-980
Enter the Price1234
Enter the Medicine Type Safe/Unsafe: Unsafe
Enter the Medicine Name:Crocin
Enter the Price1092
Enter the Medicine Type Safe/Unsafe: Safe
['M_type', 'M_name', 'M_price']
['Safe', 'Dolo-650', '12']
['Unsafe', 'Dolo-980', '1234']
['Safe', 'Crocin', '1092']
```

Question 3

From the CSV file created in the above program “Medicine.csv”
a) Display only those records where type is “unsafe”
b) Display only those records M_price is more than 400
c) To display the number of records in the file(don’t count the header row).
delimiter.

Source Code:

```
import csv
def unsafe_disp():
    with open('Medicine.csv', mode='r') as file:
        reader = csv.reader(file)
        data=list(reader)
        print(data[0])
        for row in data[1:]:
            if row[0].lower()=='unsafe':
                print(row)
            else:
                pass
```

```
def price400():
    with open('Medicine.csv', mode='r') as file:
        reader = csv.reader(file)
        data=list(reader)
        print(data[0])
        for row in data[1:]:
            if int(row[2])>400:
                print(row)

            else:
                pass
def countrow():
    with open('Medicine.csv', mode='r') as file:
        reader = csv.reader(file)
        data1=list(reader)
        data1=data1[1:]
        print('Number of Records:',len(data1))
while True:
    print('''Menu:
        1)Display only those records where type is “unsafe”
        2)Display only those records M_price is more than
400
        3)To display the number of records in the file
        4)Exit''')
    ch=int(input("Enter the choice Number:"))
    if ch==1:
        unsafe_disp()
    elif ch==2:
        price400()
    elif ch==3:
        countrow()
    elif ch==4:
        print('Exiting....')
```

```
        break
    else:
        print('Invalid Input Enter Valid Input')
```
Output:

```
Menu:
        1)Display only those records where type is “unsafe”
        2)Display only those records M_price is more than
400
        3)To display the number of records in the file
        4)Exit
Enter the choice Number:1
['M_type', 'M_name', 'M_price']
['Unsafe', 'Dolo-980', '1234']
Menu:
        1)Display only those records where type is “unsafe”
        2)Display only those records M_price is more than
400
        3)To display the number of records in the file
        4)Exit
Enter the choice Number:2
['M_type', 'M_name', 'M_price']
['Unsafe', 'Dolo-980', '1234']
['Safe', 'Crocin', '1092']
Menu:
        1)Display only those records where type is “unsafe”
        2)Display only those records M_price is more than
400
        3)To display the number of records in the file
        4)Exit
Enter the choice Number:3
Number of Records: 3
Menu:
        1)Display only those records where type is “unsafe”
        2)Display only those records M_price is more than
400
        3)To display the number of records in the file
        4)Exit
```

```
Enter the choice Number:5
Invalid Input Enter Valid Input
Menu:
        1)Display only those records where type is “unsafe”
        2)Display only those records M_price is more than
400
        3)To display the number of records in the file
        4)Exit
Enter the choice Number:4
Exiting....
```

Question 4

Write a Python program to copy the contents of above file “Medicine.csv” into “Temp.csv”, but with a different

Source Code:

```
def read_copy():
    with open ('Medicine.csv','r') as read_file:
        reader=csv.reader(read_file)
        main_data=list(reader)
    with open ('Temp.csv','w',newline='') as write_file:
        writer=csv.writer(write_file,delimiter=':')
        writer.writerows(main_data)
    with open ('Temp.csv','r') as temp_file:
        read=csv.reader(temp_file)
        for i in read:
            print(i)
read_copy()
```

Output:

```
['M_type:M_name:M_price']
['Safe:Dolo-650:12']
['Unsafe:Dolo-980:1234']
['Safe:Crocin:1092']
```

Question 5

Write a program that reads a csv file and creates another csv file with the same content except those records where the type is “safe”

Source Code:

```
import csv
def filter_safe_records():
    with open('Medicine.csv', mode='r') as file:
        reader = csv.reader(file)
        data = [row for row in reader if row[0].lower() != 'safe']
    with open('Unsafe_Medicine.csv', mode='w', newline='') as file:
        writer = csv.writer(file)
        writer.writerows(data)
    with open('Unsafe_Medicine.csv', mode='r') as file:
        read = csv.reader(file)
        for i in read:
            print(i)
filter_safe_records()
```

Output:

```
['M_type', 'M_name', 'M_price']
['Unsafe', 'Dolo-980', '1234']
```

Question 6

Read the records from "Medicine.csv" file and display alternate records

Source Code:

```
import csv
def alt_rec():
    with open ('Medicine.csv','r') as file:
        read=csv.reader(file)
        data=list(read)
        top=data[0]
        data=data[1:]
        print(top)
        for i in range(0,len(data),2):
            print(data[i])
alt_rec()
```

Output:

```
['M_type', 'M_name', 'M_price']
['Safe', 'Dolo-650', '12']
['Safe', 'Crocin', '1092']
```

Question 7

Write a program that reads a csv file and creates another csv file with price field value Increased by 20% for all the medicines where the type is unsafe

Source Code:

```
import csv
def update_unsafe_records():
    with open('Medicine.csv', mode='r') as file:
        reader = csv.reader(file)
        data = []
        for row in reader:
            if row[0].lower() == 'unsafe':
                row[2] = str(float(row[2]) * 1.2)
            data.append(row)
            print(row)
    with open('Updated_Medicine.csv', mode='w',
newline='') as file:
        writer = csv.writer(file)
        writer.writerows(data)
update_unsafe_records()
```

Output:

```
['M_type', 'M_name', 'M_price']
['Safe', 'Dolo-650', '12']
['Unsafe', 'Dolo-980', '1480.8']
['Safe', 'Crocin', '1092']
```

Data Structures

Question 1

1)Write a Menu Based Program to implement a stack that holds list of numbers. Take care of overflow/underflow situations. The Menu should be

1. Push
2. Pop
3. Display entire stack content
4. Display Top Node
5. Exit

Source Code:

```
def push(stack, value):
    if len(stack) < 10:
        stack.append(value)
    else:
        print("Stack Overflow")

def pop(stack):
    if not is_empty(stack):
        return stack.pop()
    else:
        print("Stack is empty")

def display(stack):
    return stack

def top(stack):
    if not is_empty(stack):
        return stack[-1]
    else:
        print("Stack is empty")

def is_empty(stack):
    return len(stack) == 0

# Usage:
```

```
stack = []

while True:
    print("\nMENU:")
    print("1. Push")
    print("2. Pop")
    print("3. Display")
    print("4. Display Top Node")
    print("5. Exit")

    choice = int(input("Enter your choice: "))

    if choice == 1:
        value = int(input("Enter the number to push: "))
        push(stack, value)
    elif choice == 2:
        print("Popped element:", pop(stack))
    elif choice == 3:
        print("Stack content:", display(stack))
    elif choice == 4:
        print("Top node:", top(stack))
    elif choice == 5:
        break
    else:
        print("Invalid choice. Please try again.")
```

Output:

```
Enter the Max Value of Stack:2

MENU:
1. Push
2. Pop
3. Display
4. Display Top Node
5. Exit
Enter your choice: 1
Enter the number to push: 1
```

```
MENU:
1. Push
2. Pop
3. Display
4. Display Top Node
5. Exit
Enter your choice: 1
Enter the number to push: 2

MENU:
1. Push
2. Pop
3. Display
4. Display Top Node
5. Exit
Enter your choice: 1
Enter the number to push: 3
Stack Overflow

MENU:
1. Push
2. Pop
3. Display
4. Display Top Node
5. Exit
Enter your choice: 2
Popped element: 2

MENU:
1. Push
2. Pop
3. Display
4. Display Top Node
5. Exit
Enter your choice: 3
Stack content: [1]

MENU:
1. Push
2. Pop
3. Display
4. Display Top Node
5. Exit
Enter your choice: 4
Top node: 1
```

```
MENU:
1. Push
2. Pop
3. Display
4. Display Top Node
5. Exit
Enter your choice: 2
Popped element: 1

MENU:
1. Push
2. Pop
3. Display
4. Display Top Node
5. Exit
Enter your choice: 2
Stack Underflow

MENU:
1. Push
2. Pop
3. Display
4. Display Top Node
5. Exit
Enter your choice: 7
Invalid choice. Please try again.

MENU:
1. Push
2. Pop
3. Display
4. Display Top Node
5. Exit
Enter your choice: 5
```

Question 2

Write a menu driven python program to implement operations on STACK named L that stores a list of numbers. Define the required functions as per the specifications given below

a) def PUSHSTACK (STK, N): where the numeric value N is pushed into the stack. If the size of the stack is 10 report STACK OVERFLOW situation.

b) def IS_EMPTY (STK): returns True if stack is empty otherwise returns False

c) def POPSTACK (STK): where, the function returns the value deleted from the stack.

d) def SHOWSTACK(STK): displays the contents of the stack

e) def PEEK(STK): displays the topmost element of the stack

Write the required top-level statements and function call statements. Note the POPSTACK, SHOWSTACK and PEEK functions will invoke IS_EMPTY function to check and report STACK UNDERFLOW situation.

Source Code:

```
stackmax=int(input("Enter the max value of the stack:"))
def PUSHSTACK(STK, N):
    if len(STK) < stackmax:
        value = int(input("Enter the number to push: "))
        STK.append(value)
    else:
        print("STACK OVERFLOW")
def IS_EMPTY(STK):
    return len(STK) == 0
def POPSTACK(STK):
    if not IS_EMPTY(STK):
        return STK.pop()
    else:
        print("STACK UNDERFLOW")
def SHOWSTACK(STK):
    return STK
def PEEK(STK):
    if not IS_EMPTY(STK):
```

```
        return STK[-1]
    else:
        print("Stack is empty")
L = []
while True:
    print("\nMENU:")
    print("1. Push")
    print("2. Pop")
    print("3. Display")
    print("4. Display Top Node")
    print("5. Exit")

    choice = int(input("Enter your choice: "))

    if choice == 1:
        PUSHSTACK(L, 10)
    elif choice == 2:
        print("Is Empty:", IS_EMPTY(L))
        print("Popped element:", POPSTACK(L))
    elif choice == 3:
        print("Is Empty:", IS_EMPTY(L))
        print("Stack Content:", SHOWSTACK(L))
    elif choice == 4:
        print("Is Empty:", IS_EMPTY(L))
        print("Peek:", PEEK(L))
    elif choice == 5:
        break
    else:
        print("Invalid choice. Please try again.")
```

Output:

```
Enter the max value of the stack:2

MENU:
1. Push
2. Pop
```

```
3. Display
4. Display Top Node
5. Exit
Enter your choice: 1
Enter the number to push: 1

MENU:
1. Push
2. Pop
3. Display
4. Display Top Node
5. Exit
Enter your choice: 1
Enter the number to push: 2

MENU:
1. Push
2. Pop
3. Display
4. Display Top Node
5. Exit
Enter your choice: 2
Is Empty: False
Popped element: 2

MENU:
1. Push
2. Pop
3. Display
4. Display Top Node
5. Exit
Enter your choice: 3
Is Empty: False
Stack Content: [1]

MENU:
1. Push
2. Pop
3. Display
4. Display Top Node
5. Exit
Enter your choice: 4
Is Empty: False
Peek: 1

MENU:
```

```
1. Push
2. Pop
3. Display
4. Display Top Node
5. Exit
Enter your choice: 2
Is Empty: False
Popped element: 1

MENU:
1. Push
2. Pop
3. Display
4. Display Top Node
5. Exit
Enter your choice: 2
Is Empty: True
STACK UNDERFLOW
Popped element: None

MENU:
1. Push
2. Pop
3. Display
4. Display Top Node
5. Exit
Enter your choice: 6
Invalid choice. Please try again.

MENU:
1. Push
2. Pop
3. Display
4. Display Top Node
5. Exit
Enter your choice: 5
```

Question 3

Write a Menu Based Program to implement a stack that holds details of books. Take care of overflow/underflow situations.
The Menu should be

1. Push
2. Pop
3. Display entire stack content
4. Display Top Node
5. Exit

Each node of the Stack will have the following structure
[BOOK_ID, TITLE, AUTHOR, PRICE]

Source Code:

```
stkmx=int(input("Enter the max number of books:"))
def push_book(stack, book_details):
    if len(stack) < stkmx:
        book_id = input("Enter Book ID: ")
        title = input("Enter Title: ")
        author = input("Enter Author: ")
        price = float(input("Enter Price: "))
        stack.append([book_id, title, author, price])
    else:
        print("Stack Overflow")
def pop_book(stack):
    if not is_empty(stack):
        return stack.pop()
    else:
        print("Stack Underflow")
def display_books(stack):
    return stack
def top_book(stack):
    if not is_empty(stack):
        return stack[-1]
    else:
        print("Stack is empty")
def is_empty(stack):
```

```
    return len(stack) == 0
book_stack = []
while True:
    print("\nMENU:")
    print("1. Push Book")
    print("2. Pop Book")
    print("3. Display")
    print("4. Display Top Node")
    print("5. Exit")
    choice = int(input("Enter your choice: "))
    if choice == 1:
        push_book(book_stack, [])
    elif choice == 2:
        print("Popped book:", pop_book(book_stack))
    elif choice == 3:
        print("Stack content:", display_books(book_stack))
    elif choice == 4:
        print("Top node:", top_book(book_stack))
    elif choice == 5:
        break
    else:
        print("Invalid choice. Please try again.")
```

Output:

```
Enter the max number of books:2

MENU:
1. Push Book
2. Pop Book
3. Display
4. Display Top Node
5. Exit
Enter your choice: 1
Enter Book ID: 12
Enter Title: Hello
Enter Author: Hi
Enter Price: 120

MENU:
```

```
1. Push Book
2. Pop Book
3. Display
4. Display Top Node
5. Exit
Enter your choice: 1
Enter Book ID: 1029
Enter Title: HjK
Enter Author: Hi
Enter Price: 132

MENU:
1. Push Book
2. Pop Book
3. Display
4. Display Top Node
5. Exit
Enter your choice: 1
Stack Overflow

MENU:
1. Push Book
2. Pop Book
3. Display
4. Display Top Node
5. Exit
Enter your choice: 2
Popped book: ['1029', 'HjK', 'Hi', 132.0]

MENU:
1. Push Book
2. Pop Book
3. Display
4. Display Top Node
5. Exit
Enter your choice: 3
Stack content: [['12', 'Hello', 'Hi', 120.0]]

MENU:
1. Push Book
2. Pop Book
3. Display
4. Display Top Node
5. Exit
Enter your choice: 4
Top node: ['12', 'Hello', 'Hi', 120.0]
```

```
MENU:
1. Push Book
2. Pop Book
3. Display
4. Display Top Node
5. Exit
Enter your choice: 2
Popped book: ['12', 'Hello', 'Hi', 120.0]

MENU:
1. Push Book
2. Pop Book
3. Display
4. Display Top Node
5. Exit
Enter your choice: 2
Stack Underflow
Popped book: None

MENU:
1. Push Book
2. Pop Book
3. Display
4. Display Top Node
5. Exit
Enter your choice: 6
Invalid choice. Please try again.

MENU:
1. Push Book
2. Pop Book
3. Display
4. Display Top Node
5. Exit
Enter your choice: 5
```

Question 4

Write the definition of a function POP_PUSH (LPop, LPush, N) in Python. The function should Pop out the 1ast N elements of the list LPop and Push them into the list LPush .For example:If the contents of the list LPop are [10, 15, 20, 30]And value of N passed is 2,then the function should create the list LPush as [30, 20] And the list LPop should now contain [10, 15]

NOTE : If the value of N is more than the number of elements present in LPop, then display the message "Pop not possible" .

Write the required top-level statements and function call statements and execute the program.

Source Code:

```
def POP_PUSH(LPop, LPush, N):
    if N <= len(LPop):
        LPush.extend(LPop[-N:])
        del LPop[-N:]
    else:
        print("Pop not possible")
LPop=[]
for i in range(int(input("Enter the Number of elements to
Enter into Lpop:"))):
    ele=int(input("Enter the number to enter into the Lpop:"))
    LPop.append(ele)
LPush = []
N = int(input("Enter N: "))
POP_PUSH(LPop, LPush, N)
print("LPop:", LPop)
print("LPush:", LPush)
```

Output:

```
Enter the Number of elements to Enter into Lpop:2
Enter the number to enter into the Lpop:1
Enter the number to enter into the Lpop:2
Enter N: 1
LPop: [1]
LPush: [2]
```