D D M M Y Y Y Y

EXERCISE - 10

PROGRAMS ON OOPS AND EXCEPTIONS

1. Create a class matrix. Provide a constructor which takes the dimension of the matrix. and fills with 0. Provide a function to read and populate matrix and add two matrices and display.

```
class Matrix:
    def init(self, l, b):
        self.mat = [[0 for i in range(b)] for j in
                                    range(l)]
        self.length = l
        self.breadth = b

def read(self, name):
    print("\n" + name)
    for i in self.mat:
        for j in i:
            print(j,end = " ")
        print()
    print()

def populate(self, name):
    print("Enter the elements in matrix:")
    for i in range(len(self.mat)):
        for j in range(len(self.mat[i])):
```

```
            self.mat[i][j] = int(input())
        self.read(name)

    def add(self, matA, name):
        if ((self.length != matA.length) and
                (self.breadth != A.breadth)):
            print("Error: Unequal matrix")
            return
        for i in range(len(self.mat)):
            for j in range(len(self.mat[i])):
                self.mat[i][j] += matA.mat[i][j]
        self.read(name)

l,b = input("Enter length and breadth of matrix"
                ).split(",")
l = int(l)
b = int(b)
MatA = Matrix(l,b)
MatA.populate("Matrix A:")
l,b = int(input("Enter length and breadth of
            matrix:").split(",")
MatB = Matrix(l,b)
MatB.populate("Matrix B")
MatA.add(MatB, "Matrix A")
```

2. Create a class bill. Bill should contain date, customer name and details of # of items - name, rate, quantity, total amount. Make a bill and diplay it.

```
class Item:
    def init(self, name):
        self.name = name
        self.rate = int(input("Enter rate:"))
        self.quantity = int(input("Enter quantity:"))
        self.item_total = self.rate * self.quantity
    def Display(self):
        print("Item name:", self.name)
        print("Item rate:", self.rate)
        print("Item quantity:", self.quantity)
        print("Item cost:", self.item_total)
    def Quantity_Update(self, Extra):
        self.quantity += Extra
        self.item_total = self.rate * self.quantity

class Bill:
    def init(self, cust_name, No_product):
        self.cust_name = cust_name
        self.date = input("Enter Date(dd/mm/yy):")
        self.No_product = No_product
        self.items = { }
        count = 0
        while count < No_product
            name = input("\n Enter product name:")
```

D D M M Y Y Y Y

```
if name in self.items:
    choice = input("Item already added,
                    go back? (y/n):")
    if choice == 'Y' or choice == 'y':
        pass
    else:
        print("1. Overwrite ?\n 2. Add to the
                                    quantity ?")
        choice = int(input("Enter choice:"))
        if choice == 1:
            self.items[name] = item[name]
        elif choice == 2:
            extra = int(input("Number to be
                added to previous quantity:"))
            self.items[name].Quantity_Update(extra)
        else:
            print("Error-No such option")
    else:
        self.items[name] = item(name)
        count+ = 1
self.Total_items = 0
self.total = 0
for i in self.items:
    self.total += self.items[i].item_total
    self.Total_items += self.items[i].quantity

def Display(self):
    print("\n Customer name:", self.cust_name)
```

```
print("Date of purchase", self.date)
    for i in self.items:
        self.items[i].Display()
        print()
print("Total number of items:", self.Total_items)
print("Total amount:", self.total)

cust = []
number = int(input("Enter number of customers:"))
for i in range(number):
    name = input("Enter name of customer:")
    No_products = int(input("Enter total number of
                             products:"))
print("-----BILL------")
for i in cust:
    i.Duplay()
    print("-------------")
```

D D M M Y Y Y Y

3. Write a program to handle an exception when user try to write contents to a file which is created in read mode.

```
import io
try:
    f = open("test.txt")
    f.writelines(input("Enter what you want to
                        add:"))
    f.close()

except io.UnsupportedOperation as e:
    print("Unsupported operation:", e)
except Exception as e:
    print(e)
finally:
    print("program ended")
```

Output:

Enter what you want to add: bye

Unsupported operation: not writable

program ended

4. Write a program to find reciprocal of element in given list, l = ['a', 0, 2] and handle the exception when you are finding reciprocal for the values 0 and 'a'.

```
l = ['a', 0, 2]
for i in l:
    try:
        print("reciprocal is :", 1/i)
    except ZeroDivisionError as e:
        print("cannot divide by zero :", e)
    except TypeError as e:
        print("Only numbers can be divided", e)
    except Exception as e:
        print(e)
    finally:
        print("Iteration done\n")
```

Output:

Only numbers can be divided unsupported operand type(s) for /: 'int' and 'str'
Iteration done

Cannot divide by zero: division by zero
Iteration done

reciprocal is : 0.5
Iteration done