```
import sys
from Individual import individual
from Family import family
import datetime
from prettytable import PrettyTable

validTags = ["INDI", "NAME", "SEX", "BIRT", "DEAT", "FAMC", "MARR", "HUSB",
             "WIFE", "CHIL", "DIV", "DATE", "HEAD", "TRLR", "NOTE", "FAMS",
             "FAM"]
individuals = {}
families = {}

# takes in the date string used in a Gedcom file and returns a date object of that date
def getDate(dateString):
    monthDict = {"JAN":1, "FEB":2, "MAR":3, "APR":4, "MAY":5, "JUN":6,
                 "JUL":7, "AUG":8, "SEP":9, "OCT":10, "NOV":11, "DEC":12}
    date = dateString.split()
    day = int(date[0])
    month = monthDict[date[1]]
    year = int(date[2])
    return datetime.datetime(year, month, day).date()

# takes in a Gedcom tag and returns if it is valid
def isTagValid(tag):
    return tag in validTags

# takes in a list of Gedcom rows pertaining to an individual and returns an indvidual object
def readIndividual(rowList):
    newIndiv = individual()

    # Booleans for tracking if we are reading in birth and death dates
    readingBirth = False
    readingDeath = False

    for row in rowList:
        # if the level is 1 then that means we are no longer reading in birth or death dates
        if int(row[0]) == 1:
            readingBirth = False
            readingDeath = False

        # Reads each row and sets the corresponding field in the individual object
        if row[1] == "INDI":
            newIndiv.identifier = row[2].strip()
        elif row[1] == "NAME":
            newIndiv.name = row[2].strip()
        elif row[1] == "SEX":
            newIndiv.gender = row[2].strip()
        elif row[1] == "DEAT":
            if "Y" == row[2].strip():
                # Begin reading in the death date
```

```
                readingDeath = True
                newIndiv.alive = False
            else:
                newIndiv.alive = True
        elif row[1] == "BIRT":
            # Begin reading in the birth date
            readingBirth = True
        elif int(row[0]) == 2 and row[1] == "DATE":
            if readingBirth:
                newIndiv.birthday = getDate(row[2].strip())
            elif readingDeath:
                newIndiv.deathday = getDate(row[2].strip())
        elif row[1] == "FAMS":
            newIndiv.spouseFam.append(row[2].strip())
        elif row[1] == "FAMC":
            newIndiv.childFam.append(row[2].strip())

    newIndiv.calculateAge()
    return newIndiv

# takes in a list of Gedcom rows pertaining to an family and returns an family object
def readFamily(rowList):
    newFam = family()

    # Booleans for tracking if we are reading in marriage and divorce dates
    readingMarriage = False
    readingDivorce = False

    for row in rowList:
        # if the level is 1 then that means we are no longer reading in marriage or divorce d
        if int(row[0]) == 1:
            readingMarriage = False
            readingDivorce = False

        # Reads each row and sets the corresponding field in the family object
        if row[1] == "FAM":
            newFam.identifier = row[2].strip()
        elif row[1] == "HUSB":
            newFam.husbandId = row[2].strip()
        elif row[1] == "WIFE":
            newFam.wifeId = row[2].strip()
        elif row[1] == "CHIL":
            newFam.children.append(row[2].strip())
        elif row[1] == "DIV":
            # Begin reading in the divorce date
            readingDivorce = True
            newFam.isDivorced = True
        elif row[1] == "MARR":
            # Begin reading in the marriage date
            readingMarriage = True
        elif int(row[0]) == 2 and row[1] == "DATE":
```

```
            if readingMarriage:
                newFam.married = getDate(row[2].strip())
            elif readingDivorce:
                newFam.isDivorced = True
                newFam.divorced = getDate(row[2].strip())
            if newFam.married > newFam.divorced:
              newFam.Marriagebefordivorce = True
            elif newFam.married < newFam.divorced:
              newFam.Marriagebefordivorce = False



    return newFam

# Print out the individuals and families from the Gedcom file using prettytable
def printOutput():
    indPT = PrettyTable()
    famPT = PrettyTable()

    indPT.field_names = ["ID", "NAME", "GENDER", "BIRTHDAY", "AGE", "ALIVE", "DEATH", "CHILD'
    famPT.field_names = ["ID", "MARRIED", "DIVORCED", "Married Before Divorce","Marriage befc

    for individual in sorted(individuals.keys()):
        ind = individuals[individual]
        indPT.add_row([ind.identifier, ind.name, ind.gender, ind.birthday, ind.age, ind.alive
    for family in sorted(families.keys()):
        fam = families[family]
        famPT.add_row([fam.identifier, fam.married, fam.getIsDivorced(),fam.Marriagebefordivc

    print("Individuals")
    print(indPT)
    print("Families")
    print(famPT)

# Process teh Gedcom file and store it
def processGedcomFile(file):

    # Booleans to keep track of when we are reading a family in and when we are reading an ir
    readingIndividual = False
    readingFamily = False

    # List for the lines that correspond to the family or individual we are reading in
    linesList = []

    for line in file:

        splitLine = line.split()
        # get the level for the Gedcom line
        level = splitLine.pop(0)

        # get the tag and argument for the Gedcom line
```

```
            if len(splitLine) == 1:
                tag = splitLine[0]
                arguments = ""
            else:
                if splitLine[1] == "INDI" or splitLine[1] == "FAM":
                    tag = splitLine.pop(1)
                else:
                    tag = splitLine.pop(0)

                arguments = ""
                for word in splitLine:
                    arguments = arguments + word + " "

            # Level 0 marks the start of a new family or individual
            if int(level) == 0:
                # stop reading in for the previous family or individual and add them to teh apprc
                if readingIndividual:
                    newIndividual = readIndividual(linesList)
                    individuals[newIndividual.identifier] = newIndividual
                if readingFamily:
                    newFamily = readFamily(linesList)
                    families[newFamily.identifier] = newFamily

                # start reading in for a new individual
                if tag == "INDI":
                    readingIndividual = True
                    readingFamily = False
                    linesList = []
                # start reading in for a new family
                elif tag == "FAM":
                    readingIndividual = False
                    readingFamily = True
                    linesList = []
                # stop reading in at all
                else:
                    readingIndividual = False
                    readingFamily = False

            if isTagValid(tag):
                linesList.append([level, tag, arguments])

            # Add the parents names to the families
            for family in families.keys():
              families[family].husbandName = individuals[families[family].husbandId].name
              families[family].wifeName = individuals[families[family].wifeId].name
              families[family].wddate = individuals[families[family].wifeId].deathday
              families[family].Hddate = individuals[families[family].husbandId].deathday
              if ((families[family].married < families[family].wddate) or (families[family].marri
                families[family].Marriagebedoredeath = True
              elif ((families[family].wddate == families[family].Hddate)):
                families[family].Marriagebedoredeath = True
```

```
            else:
              families[family].Marriagebedoredeath = False


def main():
   # if len(sys.argv) == 2:
        try:
            file = open('/content/Team_1_Gedcom_Project_Input.ged', "r")
            processGedcomFile(file)
            printOutput()
        except OSError:
            print("Error opening GEDCOM FILE.")

    #else:
     #   print("Error in number of arguments. Please provide the name of one GEDCOM file.")


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

Individuals

ID	NAME	GENDER	BIRTHDAY	AGE	ALIVE	DEATH	CHIL
I1	Francis /Gilbert/	M	1892-09-07	65	False	1958-08-04	N/A
I10	Larry /Honepin/	M	1954-03-13	68	True	N/A	{F2
I11	Mary /Honepin/	F	1954-03-13	68	True	N/A	{F2
I12	Lucas /Garbutt/	M	1970-11-09	51	True	N/A	{F5
I13	Micheal /Manturin/	M	1955-08-18	67	True	N/A	{F4
I14	Lily /Gilbert-Gioletti/	F	1983-05-18	39	True	N/A	{F3
I2	Bonnie /Jepson/	F	1899-09-04	65	False	1965-04-12	N/A
I3	Francine /Gilbert/	F	1927-02-14	78	False	2006-01-16	{F1
I4	Bonjour /Gilbert/	M	1928-07-17	70	False	1998-10-27	{F1
I5	Jezebel /Gilbert/	F	1938-01-13	84	True	N/A	{F1
I6	Ned /Manturin/	M	1930-07-08	55	False	1985-12-11	N/A
I7	Katherine /Gioletti/	F	1940-10-18	81	True	N/A	N/A
I8	Frank /Garbutt/	M	1938-10-30	37	False	1975-11-05	N/A
I9	Jerry /Honepin/	M	1920-04-04	68	False	1989-03-16	N/A

Families

ID	MARRIED	DIVORCED	Married Before Divorce	Marriage before death	HUSBA
F1	1926-10-15	1776-07-04	True	True	I
F2	1950-10-12	1776-07-04	True	True	I
F3	1980-09-07	1776-07-04	True	True	I
F4	1955-04-18	N/A	False	True	I
F5	1965-07-03	1776-07-04	True	True	I

Colab paid products - Cancel contracts here

✓ 0s completed at 4:18 PM