#!/usr/bin/python

"""

…

"""

import sys, locale, string

from optparse import OptionParser

class comm:

def \_\_init\_\_(self, options,file1, file2):

self.op = [options.op1, options.op2, options.op3]

self.content1 = file1.read().splitlines()

self.content2 = file2.read().splitlines()

def sorted(self):

allObj = []

attr = []

i = 0

j = 0

while i < len(self.content1) and j < len(self.content2):

if self.content1[i] == self.content2[j]:

allObj.append(self.content1[i])

attr.append(0)

del self.content1[i]

del self.content2[j]

elif self.content1[i] > self.content2[j]:

allObj.append(self.content2[j])

attr.append(2)

j += 1

else:

allObj.append(self.content1[i])

attr.append(1)

i += 1

while i < len(self.content1):

alObj.append(self.content1[i])

attr.append(1)

i += 1

while j < len(self.content2):

allObj.append(self.content2[j])

attr.append(2)

j += 1

for index in range(len(attr)):

if attr[index] == 1:

#if -1 is not on

if not self.op[0]:

print(allObj[index])

elif attr[index] == 2:

if not self.op[1]:

print('\t'\*(1-self.op[0]) + allObj[index])

else:

if not self.op[2]:

print('\t'\*(2-self.op[0]-self.op[1]) + allObj[index])

def unsorted(self):

for item in self.content1:

findSame = False

for index in range(len(self.content2)):

if item == self.content2[index]:

findSame = True

if not self.op[2]:

print('\t' \* (2-self.op[0]-self.op[1]) + item)

del self.content2[index]

break;

if not findSame:

if not self.op[0]:

print(item)

if not self.op[1]:

for item in self.content2:

print('\t'\*(1-self.op[0]) + item)

def main():

version\_msg = "%prog 1.0"

usage\_msg = """%prog [OPTION]... FILE1 FILE2

Compare two files FILE1 and FILE2 line-by-line."""

parser = OptionParser(version=version\_msg,usage=usage\_msg)

parser.add\_option("-1", action = "store\_true", dest = "op1",

default = False, help = "Suppress the output column of lines unique to file1")

parser.add\_option("-2", action = "store\_true", dest = "op2",

default = False, help = "Suppress the output column of lines unique to file2")

parser.add\_option("-3", action = "store\_true", dest = "op3",

default = False, help = "Suppress the output column of lines duplicated in file1 and file2")

parser.add\_option("-u", action = "store\_true",dest = "unsorted",

default = False, help = "compare two unsorted files")

option,args = parser.parse\_args(sys.argv[1:])

if len(args) != 2:

parser.error("wrong number of operands")

if args[0] == "-" and args[1] == "-":

parser.error("only one file can be read from input")

try:

file1 = sys.stdin if args[0] == '-' else open(args[0],'r')

file2 = sys.stdin if args[1] == '-' else open(args[1],'r')

Obj = comm(option,file1, file2)

if option.unsorted:

Obj.unsorted()

else:

Obj.sorted()

file1.close()

file2.close()

except OSError as err:

parser.error("OS error: {0}".

format(err))

if \_\_name\_\_ == "\_\_main\_\_":

main()

----------------------------------------------------------------------------------------------------

#!/usr/bin/python

import sys,locale,string

#find all objects and propositions

def find\_old\_obj\_prop():

with open("oldObjNProp","w") as outFile:

for i in range(469):

with open("./sentence-completed/%s.instenv" % i,"r") as inFile:

outFile.write('\n' + inFile.readline() + '\nProposition: ');

#find all the propositions

prop = set(inFile.read().replace('\n',',').split(','));

for elem in prop:

# if(elem.count("state") != 0):

outFile.write(elem );

outFile.write('\n\n\n');

#find all propositions involving 2 objects

def find\_2\_prop():

with open("2ObjProp","w") as outFile:

for i in range(469):

with open("./sentence-completed/%s.instenv" % i,"r") as inFile:

outFile.write('\n' + inFile.readline() + '\n');

#find all the propositions

prop = set(inFile.read().replace('\n',',').split(','));

for elem in prop:

if(elem.count(' ') ==2):

outFile.write(elem);

outFile.write('\n');

#find all propositions has "states"

def find\_state\_prop():

prop = set();

with open("stateProp","w") as outFile:

for i in range(469):

with open("./sentence-completed/%s.instenv" % i,"r") as inFile:

# outFile.write('\n' + inFile.readline() + '\n');

inFile.readline();

#find all the propositions

prop = prop | set(inFile.read().replace('\n',',').split(','));

for elem in prop:

if(elem.count("state") != 0):

outFile.write(elem);

def main():

if len(sys.argv) != 2:

print("Please specify an option!");

exit(1);

if sys.argv[1] == "old":

find\_old\_obj\_prop();

elif sys.argv[1] == "2prop":

find\_2\_prop();

elif sys.argv[1] == "stateprop":

find\_state\_prop();

else:

print("cannot recognize the command!");

exit(1);

if \_\_name\_\_ == "\_\_main\_\_":

main();