import sys

import random

def turnmove(): ##누가 선일지 정하는 함수

if len(players) == 2 or len(players) == 3:

players.append(players[0])

del players[0]

elif len(players) == 4:

for i in range(2):

players.append(players[0])

del players[0]

elif len(players) == 5:

for i in range(3):

players.append(players[0])

del players[0]

def betting():##베팅 함수

i = 0

calls = 0 ##콜 한 횟수(플레이어 수 만큼 모여야 다음으로 넘어감)

global fold

global lowlimit

while 1:

i = i%len(players)

while 1:

if players[i][2] == 0:

calls += 1

break

if table[i] != max(table) and max(table) != 2000:

if players[i][0] == 'player':

act = int(input('%s의 턴입니다.\n행동을 입력하세요.\n1 : Fold\n2 : Call\n3 : Raise\n0 : 내상태보기\n' %players[i][0]))

else:

act = players[i][4].act1(turn, players[i][5], players[i][6]) #AI행동 case1호출 - 무조건 콜

elif table[i] == max(table) and max(table) != 2000:

if players[i][0] == 'player':

act = int(input('%s의 턴입니다.\n행동을 입력하세요.\n1 : Fold\n2 : Check\n3 : Raise\n0 : 내상태보기\n' %players[i][0]))

else: #AI의 함수호출

act = players[i][4].act2(turn, players[i][5], players[i][6]) #AI - 랜덤

elif table[i] != max(table) and max(table) == 2000:

if players[i][0] == 'player':

act = int(input('%s의 턴입니다.\n행동을 입력하세요.\n1 : Fold\n2 : Call\n0 : 내상태보기\n' %players[i][0]))

else:

act = players[i][4].act3()

elif table[i] == max(table) and max(table) == 2000:

act = int(input('%s의 턴입니다.\n행동을 입력하세요.\n1 : Fold\n2 : Check\n0 : 내상태보기\n' %players[i][0]))

if act == 1:

print('Fold')

players[i][2] = 0

calls += 1

fold += 1

elif act == 2:

if table[i] != max(table):

print('콜합니다')

players[i][1] -= max(table)-table[i]

table[i] += max(table)-table[i]#남들이 베팅한 금액에 맞춰 베팅

else :

print('체크합니다')

calls += 1

elif act == 3:

if lowlimit < 2000:

if players[i][0] == 'player':

bet = int(input('베팅금액을 입력하십시오.\n 최소 배팅금액은 %d, 최대 배팅금액은 2000 ' %lowlimit))

else: #AI 베팅

print(players[i][0] + "raise:")

bet = players[i][4].bet(lowlimit, 2000)

print(bet)

if lowlimit<=bet<=2000:

players[i][1] -= bet

table[i] += bet

calls = 1

lowlimit = bet\*2

else:

print('다시 입력해 주십시오')

continue

elif 2000<=lowlimit:

if players[i][0] == 'player':

bet = int(input('베팅금액은 2000입니다. 2000 입력해주십시오. '))

else:#AI베팅

print(players[i][0] + "raise:")

bet = players[i][4].bet(lowlimit, lowlimit)

print(bet)

if bet == 2000:

players[i][1] -= bet

table[i] += bet

calls = 1

lowlimit = bet\*2

else:

print('다시 입력해 주십시오')

elif lowlimit == 4000:

continue

elif act == 0:

print(players[i])

print(pot)

print(table)

print(comcard)

continue

elif act == 9:

print(deck)

print(len(deck))

else:

continue

break

i += 1

if fold == opnum:

break

if calls == len(players):

break

def tabletopot():##테이블->팟으로 돈 이동

global pot

for i in range(len(table)):

pot += table[i]

table[i] = 0

def nodecision():

global pot

global fold

print('무승부입니다. 판돈을 나눠갖습니다.')

c = 0

for i in range(len(players)):

if players[i][2] == 44:

c += 1

for i in range(len(players)):

if players[i][2] == 44:

players[i][1] += pot/c

def showdown():

print('쇼다운')

global pot

score = [0, 0, 0, 0, 0]##쇼다운의 결과 점수만 모아놓은 리스트

maxnum = [0, 0, 0, 0, 0]##쇼다운의 결과 카드 번호만 모아놓은 리스트

maxlist = [0, 0, 0, 0, 0]## 쇼다운의 결과 같은번호의 카드 갯수를 모아놓은 리스트

maxindex = []

straightnum = [0, 0, 0, 0, 0]##스트레이트 승자판정용 리스트

flushcard = [0, 0, 0, 0, 0]

comli = [0, 0, 0, 0, 0]

for i in range(len(players)):

if players[i][2] == 0:

continue

else:

cards = []##결과물을 조합할 리스트

cards.append(players[i][5])

cards.append(players[i][6])

cards += comcard

straight = 0

flush = 0

num = [cards[j][1] for j in range(7)]

num.sort()

maxnum[i] = num

suit = [cards[j][0] for j in range(7)]

suit.sort()

li = [num.count(j) for j in range(2, 15)]

maxlist[i] = li

if 4 in li:

players[i][3] = 8

elif 3 in li and 2 in li:

players[i][3] = 7

elif 3 in li and 2 not in li:

players[i][3] = 4

elif li.count(2) == 2 or li.count(2) == 3:

players[i][3] = 3

elif li.count(2) == 1:

players[i][3] = 2

elif li.count(1) == 7:

players[i][3] = 1

if suit.count('s') >= 5 or suit.count('h') >= 5 or suit.count('c') >= 5 or suit.count('d') >= 5:

fli = []

for j in range(len(cards)):

if suit.count('s') >= 5:

if cards[j][0] == 's':

fli.append(cards[j][1])

elif suit.count('h') >= 5:

if cards[j][0] == 'h':

fli.append(cards[j][1])

elif suit.count('c') >= 5:

if cards[j][0] == 'c':

fli.append(cards[j][1])

elif suit.count('d') >= 5:

if cards[j][0] == 'd':

fli.append(cards[j][1])

fli.sort()

fli.reverse()

del fli[5:]

flushcard[i] = fli

flush = 1

if players[i][3] < 6:

players[i][3] = 6

if li[12]>0 and li[11]>0 and li[10]>0 and li[9]>0 and li[8]>0:

straight = 2

elif li[12]>0 and li[0]>0 and li[1]>0 and li[2]>0 and li[3]>0:

straight = 1

straightnum[i] = li[3]

else:

for j in range(3):

if num[6-j]-num[2-j] == 4 and len(set(num[2-j:7-j])) == 5 :

straight = 1

straightnum[i] = num[6-j]

if straight == 2 and flush == 1:

players[i][3] = 10

elif flushcard[i] != 0 and flushcard[i][4] - flushcard[i][0] == 4:

players[i][3] = 9

elif straight == 1:

players[i][3] == 5

score[i] = players[i][3]

print(players)

for i in range(len(players)):

li = []

if players[i][3] == max(score) and score.count(max(score)) == 1:

players[i][2] = 99

elif players[i][3] == max(score) and score.count(max(score)) != 1:

if players[i][3] == 10:

comli[i] = 44

elif players[i][3] == 9:

li.append(straightnum[i])

comli[i] = li

elif players[i][3] == 8:

li.append(maxlist.index(4))

while 1:

a = maxlist[i].pop()

if a != 0 and a != 4:

li.append(len(maxlist[i]))

break

comli[i] = li

elif players[i][3] == 7:

while 1:

a = maxlist[i].pop()

if a == 3:

li.append(len(maxlist[i]))

break

if 3 in maxlist[i]:

li.append(maxlist[i].index(3))

else:

while 1:

a = maxlist[i].pop()

if a == 2:

li.append(len(maxlist[i]))

break

comli[i] = li

elif players[i][3] == 6:

comli[i] = flushcard[i][0:5]

elif players[i][3] == 5:

li.append(straightnum[i])

comli[i] = li

elif players[i][3] == 4:

li.append(maxlist[i].index(3))

c = 0

while 1:

a = maxlist[i].pop()

if a == 1:

li.append(len(maxlist[i]))

c += 1

if c == 2:

break

comli[i] = li

elif players[i][3] == 3:

for j in range(2):

li.append(maxlist[i].index(2))

maxlist[i][maxlist[i].index(2)] = 0

while 1:

a = maxlist[i].pop()

if a == 1:

li.append(len(maxlist[i]))

break

comli[i] = li

elif players[i][3] == 2:

li.append(maxlist[i].index(2))

c = 0

while 1:

a = maxlist[i].pop()

if a == 1:

li.append(len(maxlist[i]))

c += 1

if c == 3:

break

comli[i] = li

elif players[i][3] == 1:

c = 0

while 1:

a = maxlist[i].pop()

if a == 1:

li.append(len(maxlist[i]))

c += 1

if c == 5:

break

comli[i] = li

for i in range(len(comli)):

if comli[i] != 0:

a = len(comli[i])

print(comli)

if comli.count(0) < 4:

for k in range(a):

if comli.count(0) == 4:

break

prime = 0

li = []

for i in range(len(comli)):

if comli[i] != 0 and comli[i] != 44:

li.append(comli[i][k])

prime = max(li)

for i in range(len(comli)):

if comli[i] != 0 and comli[i] != 44:

if comli[i][k] < prime:

comli[i] = 0

if comli.count(0) < 4:

for i in range(len(comli)):

if comli[i] != 0:

comli[i] = 44

for i in range(len(comli)):

if comli[i] == 44:

nodecision()

comli = [0, 0, 0, 0, 0]

break

print(comli)

for i in range(len(comli)):

if comli[i] != 0:

players[i][2] = 99

print(players)

for i in range(len(players)):

if players[i][2] == 99:

if players[i][3] == 10:

print('%s가 Royal Flush로 승리하였습니다!' %players[i][0])

elif players[i][3] == 9:

print('%s가Straight Flush로 승리하였습니다!' %players[i][0])

elif players[i][3] == 8:

print('%s가 Four of a Kind로 승리하였습니다!' %players[i][0])

elif players[i][3] == 7:

print('%s가 Full House로 승리하였습니다!' %players[i][0])

elif players[i][3] == 6:

print('%s가 Flush로 승리하였습니다!' %players[i][0])

elif players[i][3] == 5:

print('%s가 Straight로 승리하였습니다!' %players[i][0])

elif players[i][3] == 4:

print('%s가 Three of a Kind로 승리하였습니다!' %players[i][0])

elif players[i][3] == 3:

print('%s가 Two Pair로 승리하였습니다!' %players[i][0])

elif players[i][3] == 2:

print('%s가 One Pair로 승리하였습니다!' %players[i][0])

elif players[i][3] == 1:

print('%s가 High Card로 승리하였습니다!' %players[i][0])

players[i][1] += pot

pot = 0

#버그목록 : 원페어인데 투페어로 판정나는 부분, 같은 족보에서 높은카드 순서대로 점수가 매겨지는게 안되는 부분, 특정 상황에서 승패가 갈리지 않고 넘어가는 부

while 1:

opnum = int(input('상대의 숫자?(0 : 게임종료, 최대 4)'))

if opnum == 0:

sys.exit()

elif 0 < opnum < 5:

break ##상대의 수 정하기

player = ['player', 50000, 1, 0, '주소']##플레이어 초기화, 50000은 자본금, 1은 폴드가 아님을 표시, 0은 승패용 점수

##주소는 객체만 필요하지만 인덱스 수를 맞추기 위해 추가

players = [player]

for i in range(1, opnum+1):

globals()['opponent{}'.format(i)] = [str('opponent{}'.format(i)), 50000, 1, 0, '주소']##opponent1, 2, 3과 같은 이름으로 상대방 초기화, 객체주소위치

players.append(globals()['opponent{}'.format(i)])

##AI클래스@@@@@@@@@@@@

class AIplayer:

fold = 0

def name(self, name):

self.name = name

return self.name

def act1(self, turn, hand1, hand2):

#print("(act1)"+"turn:"+str(turn)+ " ")

print(self.name+":")

if turn == 0: #turn0 일때 핸드체크

if self.hand\_check(hand1, hand2) == 'Raise':

return 3

elif self.hand\_check(hand1, hand2) =='Fold':

return 1

return 2

def act2(self, turn, hand1, hand2):

#print("(act2)"+"turn:"+str(turn)+ " ")

print(self.name+":")

if turn == 0: #turn 0일 때 핸드체크

if self.hand\_check(hand1, hand2) == 'Raise':

return 3

elif self.hand\_check(hand1, hand2) =='Fold':

return 1

return random.randrange(1,4)

def act3(self): #랜덤

#print("(case3)"+"turn:"+str(turn)+ " "

print(self.name+":")

return random.randrange(1,3)

def bet(self, lowlimit, maxlimit): #베팅함수

return random.randrange(lowlimit, maxlimit+1)

def showhand(self): #패공개

return random.randrange(1,5)

def hand\_check(self, hand1, hand2): #핸드체크

# 프레플랍 단계 먼저 자신에게 주어진 카드 정보를 분석한다.

premium = [('p', 14,14),('p',13,13),('p',12,12),('s',14,13),('p',11,11),('s',14,12),('s',13,12),('s',14,11),('s',13,11),('p',10,10),('o',14,13),('s',14,10),('s',12,11),('s',13,10),('s',12,10),('s',11,10),('p',9,9)]

special = [('o',14,12),('s',14,9),('o',13,12),('p',8,8),('s',13,9),('s',10,9),('s',14,8),('s',12,9),('s',11,9),('o',14,11),('s',14,5),('p',7,7),('s',14,7),('o',13,11),('s',14,4),('s',14,3),('s',14,6)]

good = [('o',12,11),('p',6,6),('s',13,8),('s',10,8),('s',14,2),('s',9,8),('s',11,8),('o',14,10),('s',12,8),('s',13,7),('o',13,10),('p',5,5),('o',11,10),('s',8,7),('o',12,10),('p',4,4),('p',3,3),('p',2,2)]

# 숫자 비교 후 pair인지 확인(pair이면 p, 아니면 문자 확인으로 넘어감), 문자 확인해서 일치하는지 확인(불일치 하면 o, 일치하면 s ), 숫자 조합해서 하나의 튜플로 제작

if hand1[1] == hand2[1]:

tup1 = 'p'

else:

if hand1[0] == hand2[0]:

tup1 = 's'

else:

tup1 = 'o'

if hand1[1] > hand2[1]: # 숫자 큰 게 앞으로 오게 함

tup2, tup3 = hand1[1], hand2[1]

else:

tup2, tup3 = hand2[1], hand1[1]

tup = (tup1,tup2,tup3) #튜플 생성

#print(tup) #확인용

if premium.count(tup) == 1 or special.count(tup) == 1 or good.count(tup) == 1: #판단

return ('Raise')

else:

return ('Fold')

##객체생성@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

for i in range(1, opnum+1):

name = players[i][0]

players[i][4] = AIplayer()

players[i][4].name(name) #이름 저장

count = 0##게임 횟수

while 1:

fold = 0#포기한 사람의 수

global turn #턴 카운터

turn = 0

comcard = []#공용패

pot = 0##확정 판돈

table = [] #베팅금 임시보관

for i in range(len(players)):

table.append(0)

count += 1

deck = [(suit, i) for suit in ["s", "h", "d", "c"] for i in range(2,15)]##카드 덱 만들기, 튜플로 구성된 리스트

random.shuffle(deck)

if len(players) > 2:

print('딜러버튼은 %s입니다. 스몰블라인드 빅블라인드 지불합니다.' %players[0][0])

players[1][1] -= 100

table[1] += 100

players[2][1] -= 200

table[2] += 200

else:

print('딜러버튼은 %s입니다. 블라인드 지불합니다.' %players[0][0])

players[1][1] -= 200

table[1] += 200

players.append(players[0])

del players[0]

table.append(table[0])

del table[0]

for i in range(len(players)\*2):##각각의 플레이어가 [이름, 자본금, 폴드여부, (카드1), (카드2)]의 모양으로 손패를 가지게 됨

players[i%len(players)].append(deck.pop())

if len(players) > 2:##플레이어 숫자에에 따라 순서 정하기

for i in range(2):

players.append(players[0])

del players[0]

table.append(table[0])

del table[0]

else:

players.append(players[0])

del players[0]

table.append(table[0])

del table[0]

print(table)

while 1:

print(players)

if turn == 0:

lowlimit = 400

else:

lowlimit = 200

betting()

tabletopot()

if fold == opnum:##1명 빼고 모두가 폴드한 경우

for i in range(len(players)):

if players[i][2] == 1:

tabletopot()

players[i][1] += pot

pot = 0

if players[i][0] == 'player':

showhand = int(input('패를 공개하시겠습니까?\n1. 전부 공개\n2.%s만 공개\n3.%s만 공개\n그 외 키 : 공개안함\n' %(players[i][5], players[i][6])))

else:#AI 패공개

print('패를 공개하시겠습니까?\n1. 전부 공개\n2.%s만 공개\n3.%s만 공개\n그 외 키 : 공개안함\n' %(players[i][5], players[i][6]))

showhand = players[i][4].showhand()

print(str(showhand)+ '선택')

if showhand == 1:

print(players[i][5], players[i][6])

elif showhand == 2:

print(players[i][5])

elif showhand == 3:

print(players[i][6])

else:

print('패 공개 안함')

if turn == 0:

turnmove()

break

turn += 1

if turn == 1:

for i in range(3):##카드 3장 깔기

comcard.append(deck.pop())

turnmove()

elif turn < 4:##턴이 2, 3 일때 카드 1장씩 깔기

comcard.append(deck.pop())

elif turn == 4: #쇼다운으로 넘어감

showdown()

break

print(comcard)

for i in range(len(players)):

players[i][2] = 1 ##폴드한 사람들 부활

del players[i][5:7] ##손패 초기화

players[i][3] = 0 ##점수초기화