import random

def function\_1():

file\_1 = open('input\_file.txt', 'r')

a, run\_target = list(map( int, file\_1.readline().split() ))

list\_of\_players = []

average\_run = []

for temp in file\_1.readlines():

temp = temp.strip().split(' ')

if len(temp) == 1:

store = 0

while store < len(temp[0]):

if temp[0][store].isalpha():

store =store+ 1

else:

break

average\_run.append(int(temp[0][store: ]))

list\_of\_players.append(temp[0][0: store])

if len(temp) == 2:

average\_run.append(int(temp[1]))

list\_of\_players.append(temp[0])

file\_1.close()

return a, run\_target, list\_of\_players, average\_run

#------------------------------------------

def function\_2():

master = []

while len(master) < 15:

new\_master = []

for k in range(a):

new\_master.append(random.randint(0, 1))

if (new\_master not in master) and (0< sum(new\_master)) :

master.append(new\_master)

return master

#----------------------------------------

def fitness\_function(people):

total = 0

for k in range(len(people)):

if people[k]:

total =total+ average\_run[k]

return abs(run\_target - total)

def crossover\_function(master1, master2):

cross\_point = a // 2

children1 = master1[0: cross\_point] + master2[cross\_point: ]

children2 = master2[0: cross\_point] + master1[cross\_point: ]

return children1, children2

#-----------------------------------------

def mutation\_function(people):

k = random.randint(0, a-1)

if people[k] != 0:

people[k] = 0

if people[k] == 0:

people[k] = 1

def genetic\_algorithms():

master = function\_2()

empty\_lst = []

for p in range(500):

while len(master) > 1:

record2 = random.randint(0, len(master)-1)

record1 = random.randint(0, len(master)-1)

while record1 == record2:

record2 = random.randint(0, len(master) - 1)

master2 = master[record2]

master1 = master[record1]

children1, children2 = crossover\_function (master1, master2)

#----------------------------------------

number = 0.5

if number <= random.random() :

mutation\_function(children1)

if number<= random.random():

mutation\_function(children2)

if fitness\_function(children2) == 0:

return children2

if fitness\_function(children1) == 0:

return children1

if sum(children2) == 0:

children2 = None

if sum(children1) == 0:

children1 = None

if children2:

empty\_lst.append(children2)

if children1:

empty\_lst.append(children1)

master.remove(master2)

master.remove(master1)

if len(master) == 1:

if fitness\_function(master[0]) == 0:

return master[0]

master = empty\_lst

empty\_lst = []

return -1

#------------------------------------------

if \_\_name\_\_ == '\_\_main\_\_':

a, run\_target, list\_of\_players, average\_run = function\_1()

final\_players = genetic\_algorithms()

print(list\_of\_players)

if final\_players == -1:

print(-1)

else:

print("".join(map(str, final\_players)))