import random as rand

total\_batsmen = 0

target\_score = 0

batsmen = []

with open('lab2\_v1.txt', 'r') as txt:

file = txt.readline().split()

total\_batsmen = int(file[0])

target\_score = int(file[1])

for i in range(total\_batsmen):

batsman = txt.readline().strip('\n').split()

batsmen.append(batsman)

population\_size = 10

population = []

chromosome\_length = total\_batsmen

chromosome = []

chromosome\_no = 0

while chromosome\_no < population\_size:

'''May produce less than 10 chromosomes as we are not taking any duplicate'''

chromosome = []

for i in range(chromosome\_length):

chromosome.append(rand.randint(0, 1))

if chromosome not in population:

population.append(chromosome)

chromosome\_no += 1

def fitness(population, target\_score):

chromosome\_score = []

for i in range(len(population)):

score = 0

for j in range(len(population[i])):

if population[i][j] == 1:

score += int(batsmen[j][1])

chromosome\_score.append((i, abs(target\_score - score)))

return chromosome\_score

def selection(chromosome\_score):

new\_population = []

probability\_of\_chosen = []

total\_fitness = 0

for i in range(len(chromosome\_score)):

total\_fitness += chromosome\_score[i][1]

for i in range(len(chromosome\_score)):

fitness = round(chromosome\_score[i][1] / total\_fitness \* 100)

probability\_of\_chosen.append(fitness)

total\_probability = 0

for i in range(len(probability\_of\_chosen)):

total\_probability += probability\_of\_chosen[i]

mean = round(total\_probability / len(probability\_of\_chosen))

less\_than\_mean = []

for i in range(len(probability\_of\_chosen)):

if probability\_of\_chosen[i] < mean:

less\_than\_mean.append((i, probability\_of\_chosen[i]))

for i in range(len(probability\_of\_chosen)):

if probability\_of\_chosen[i] < mean:

new\_population.append(population[i])

else:

try:

'''In case there is no element in the less\_than\_mean list'''

random\_choice = rand.choice(less\_than\_mean)

new\_population.append(population[random\_choice[0]])

except:

random\_choice = rand.randint(0, len(probability\_of\_chosen) - 1)

new\_population.append(population[random\_choice])

return new\_population

def crossover(new\_population):

children = []

i = 0

while i < len(new\_population) - 1:

crossover\_point = rand.randint(1, chromosome\_length - 1)

child\_one = new\_population[i][:crossover\_point] + new\_population[i + 1][crossover\_point:]

child\_two = new\_population[i + 1][:crossover\_point] + new\_population[i][crossover\_point:]

children.append(child\_one)

children.append(child\_two)

i += 2

return children

def mutation(children):

for i in range(len(children)):

mutate = rand.randint(0, 1)

if mutate:

index\_pos = rand.randint(1, chromosome\_length - 1)

if children[i][index\_pos] == 0:

children[i][index\_pos] = 1

else:

children[i][index\_pos] = 0

return children

def gen\_algo(population):

n = 0

while n < 1000:

chromosome\_score = fitness(population, target\_score)

new\_population = selection(chromosome\_score)

children = crossover(new\_population)

children = mutation(children)

child = []

for i in range(len(children)):

score = 0

for j in range(len(children[i])):

if children[i][j] == 1:

score += int(batsmen[j][1])

if target\_score - score == 0:

child = children[i]

return child

population = children[:]

n += 1

winning\_combo = gen\_algo(population)

players = [b[0] for b in batsmen]

print(players)

if winning\_combo is not None:

print(\*winning\_combo)

else:

print(-1)