import random

def generate\_population(size):

# I am generating 6 sample of population.

population = []

for i in range(6):

strng = ""

for j in range(size):

strng += str(random.randint(0, 1))

population.append(strng)

return population

def fitness\_calculation(population, transactions):

fitness = []

for i in range(len(population)):

total = 0

for j, k in enumerate(population[i]):

if k == '1':

total += transactions[j]

fitness.append(abs(total))

return fitness

def random\_selection(population, fitness):

# I am discarding one child with the worst fitness score

max\_index = fitness.index(max(fitness))

index = random.randint(0, len(population)-1)

while index == max\_index:

index = random.randint(0, len(population)-1)

return population[index]

def crossover(parent1, parent2):

cross\_point = random.randint(0, len(parent1)-1)

child = parent1[:cross\_point]+parent2[cross\_point:]

return child

def mutation(child):

index = random.randint(0, len(child)-1)

if child[index] == '0':

mutated\_child = child[0:index]+'1'+child[index+1:]

else:

mutated\_child = child[0:index]+'0'+child[index+1:]

return mutated\_child

def genetic\_algorithm(population, transactions, count):

if count >= 900:

return -1

fitness = fitness\_calculation(population, transactions)

for i, j in enumerate(fitness):

if j == 0 and int(population[i]) != 0:

return population[i]

new\_population = []

for i in range(len(population)):

parent1 = random\_selection(population, fitness)

parent2 = random\_selection(population, fitness)

child = crossover(parent1, parent2)

if random.random() > 0.6:

child = mutation(child)

new\_population.append(child)

return genetic\_algorithm(new\_population, transactions, count+1)

input\_file = open("D:/MSAS/10th Semester/CSE422/Labs/Lab 2/input.txt", 'r')

lines = input\_file.readlines()

transactions = []

for i in range(1, int(lines[0])+1):

t\_type, amount = lines[i].split()

if t\_type == "l":

transactions.append(-int(amount))

elif t\_type == "d":

transactions.append(int(amount))

population = generate\_population(len(transactions))

print(genetic\_algorithm(population, transactions, 0))