```
import random
import numpy as np

#Generating chromosomes
def generate_population(chromosone_numbers):
   chromosome_lst=[]
   for i in range(8): #need to upadte the range around 100
      chromosome=np.random.randint(0,2,size=chromosone_numbers)
      chromosome_lst.append(chromosome)
   #print(f'Chromosomes\n')
   #for i in chromosome_lst:
      #print(i)
   #print("__"*14)
   return chromosome_lst

#Calculating fitness for each chromosome
def fitness_calculation(chromosome_lst):
   fitness_score=[]
   for i in range(len(chromosome_lst)):
      score=0
      #Calculating fitness for each particular chromosome
      for idx, gene in enumerate(chromosome_lst[i]):
         if gene!=0:
            score+=transaction[idx]
      fitness_score.append(abs(score))
   #print(fitness_score)
   return fitness_score

#Finding the least fit chromosome and checking if the goal is achieved or not
def selection(fitness_scores):
    return fitness_scores.index(min(fitness_scores))
```

```
#Crossing over between 2 chromosome
def crossover(chr1,chr2):
    chr1,chr2=list(chr1),list(chr2)
    point=random.randint(1,len(chr1)-2)
    chr1_fh,chr1_lh=chr1[:point+1],chr1[point+1:]
    chr2_fh,chr2_lh=chr2[:point+1],chr2[point+1:]
    child1=(chr1_fh+chr2_lh)
    child2=(chr2_fh+chr1_lh)
    child_lst=[]
    child_lst.append(child1)
    child_lst.append(child2)

    return child_lst

#Picking a random child and mutating their gene in the chromosome
def mutation(off1,off2):
    #print(f"offspring1: {off1}\n offrping 2: {off2}")
    #choosing a random point to mutate
    point=random.randint(0,len(off1)-1)
    #print('mutaion point', point)

    if off1[point]==0:
        off1[point]=1
    else:
        off1[point]=0
        #print('mutated off 1',off1)

    if (off2[point]==0):
        off2[point]=1
    else:
        off2[point]=0
        #print('mutated off 2',off2)

    return off1,off2



transaction=[]
goal_achieved=False
output_sequence=np.array([])#Output list for initialization
inp_file=open('Input_file_02.txt','r')
lines=inp_file.readlines()
number_of_transaction=int(lines[0])
inp_file.close()
for i in range(1,len(lines)):
    transaction_type,amount=lines[i].split()
    if transaction_type=='l':
        transaction.append(-int(amount))
    else:
        transaction.append(int(amount))
#print(f'Trsancation taken place: {transaction}')

#loop for Genetic Algorithm
for i in range(1500): #need to upadte the range around 150
    #generating population
    population=generate_population(number_of_transaction)
    #finding the fitnesss of each chromose in the population
    fitness=fitness_calculation(population)

    #Finding the least_fit chromosome index and checking if that satisfies the condition
    x = selection(fitness)
    if fitness[x] == 0:
```

```
 if list(population[x]) != [0]*number_of_transaction:
  #print(i,j)
  goal_achieved=True
  output_sequence=np.array(population[x])
  break

for j in range(len(population)):
  #randomly choosing two chromose for crossing over
  chr1=random.choice(population)
  chr2=random.choice(population)
  offspr=crossover(chr1,chr2)
  offspr1,offspr2=offspr[0],offspr[1]

  mutated=mutation(offspr1,offspr2)

  #Checking if anyof the mutated chromosome help us achieve the target balance
  total_balance=0
  for x,y in enumerate(offspr1):
    if x==1:
      total_balance+=transaction[x]
  if total_balance==0:
    if offspr1!=([0] * len(offspr1)):
      output_sequence=np.array(offspr1)
      goal_achieved=True
      break

  total_balance=0
  for x,y in enumerate(offspr2):
    if x==1:
      total_balance+=transaction[x]
  if total_balance==0:
```

```
            if offspr2!=([0] * len(offspr2)):

                output_sequence=np.array(offspr2)

                goal_achieved=True

                break


    #if result achieved terminate the code

    if goal_achieved==True:

        break


if output_sequence.size==0:

    #print(i,j)

    print(-1)


else:

     print(output_sequence)
```