## Cost of Ballons

- You are conducting a contest at your college. This contest consists of two problems and participants. You know the problem that a candidate will solve during the contest.
- You provide a balloon to a participant after he or she solves a problem. There are only green and purple-colored balloons available in a market. Each problem must have a balloon associated with it as a prize for solving that specific problem. You can distribute balloons to each participant by performing the following operation:
- 1.Use green-colored balloons for the first problem and purple-colored balloons for the second problem
- 2.Use purple-colored balloons for the first problem and green-colored balloons for the second problem
- You are given the cost of each balloon and problems that each participant solve. Your task is to print the minimum price that you have to pay while purchasing balloons.
- Input format
- First line: T that denotes the number of test cases (1<=T<=10)
- For each test case:
  - First line: Cost of green and purple-colored balloons
  - Second line: n that denotes the number of participants (1<=n<=10)
  - Next n lines: Contain the status of users. For example, if the value of the jth integer in the ith row is 0 , then it depicts that the ith participant has not solved the jth problem. Similarly, if the value of the jth integer in the ith row is 1 , then it depicts that the ith participant has solved the jth problem.
- Output format
  - For each test case, print the minimum cost that you have to pay to purchase balloons.
- SAMPLE INPUT - SAMPLE OUTPUT
  - t= 2
  - g = 9 p = 6 or g = 6 p = 9 --> 69
  - 10
  - 1 1
  - 1 1
  - 0 1
  - 0 0
  - 0 1
  - 0 0
  - 0 1
  - 0 1
  - 1 1
  - 0 0

---

  - g=1 p=9 or g=9 p = 1 ---> 14
  - 10
  - 0 1
  - 0 0
  - 0 0
  - 0 1

- 1 0
- 0 1
- 0 1
- 0 0
- 0 1
- 0 0

In [11]:

```python
# Cost of ballons
t= int(input())
for i in range(1,t+1):
    ballon=input().split()
    g=int(ballon[0])
    p = int(ballon[1])
    n=int(input())
    sum1 = 0
    sum2 = 0
    for i in range(1,n+1):
        probs=input().split()
        p1=int(probs[0])
        p2=int(probs[1])
        t1=g
        t2=p
        if p1==1 and p2 ==1:
            sum1 = sum1 +(t1+t2)
        elif p1==1 and p2 == 0:
            sum1 = sum1 + t1
        elif p1==0 and p2 == 1:
            sum1 = sum1 + t2
        t3=p
        t4=g
        if p1==1 and p2 ==1:
            sum2 = sum2 +(t3+t4)
        elif p1==1 and p2 == 0:
            sum2 = sum2 + t3
        elif p1==0 and p2 == 1:
            sum2 = sum2 + t4
    if(sum1>sum2):
        print(sum2)
    else:
        print(sum1)

    #print(sum1)
    #print(sum2)

```

```
1
2 3
5
1 1
1 0
1 0
0 1
1 0
14
```

## Date: 20 June 2019

## Day Objectives

- Regular Expressions
  - Constructing Regular Expressions for various use cases
  - Regular Expressions Module and related in Python
  - Improving the Contacts applications with name and phone number validation using regular expressions
- File Handling
  - Text Files
  - Upgrading the Contacts Applications to store contact information in a text file

```
#### Regular Expression

- Pattern Matching
- Symbolic Notation of a pattern
    - Pattern : Format which repeats
    - Pattern(RE) represents The set of all values that  matches that pattern
- [0-9] -> Any digit
- [a-z] -> Any lower case alphabet
- [2468] -> All single digit multiples of 2-->
     [8624]
     [6824]
     [2864]
     [6842]  -> so many ways we can define the above expression
- ^[0-9]{1}$ -> Only single digit number


- ^[0-9]{3}$ -> Only 3 digits  numbers


- [0-9]*0$    -> All multiples of 10

- ^[1-9][0-9]*0$   -->start with any number b/w 1 to 9 ends with 0


- ^([1-9][0-9]*[05])$|^([5])$   ---> All multiples of 5

- [w][o][r][d] ---> Searching for 'word'

- ^[1-9][0-9]{9}$ ---> All 10 digit numbers

-   (^[6-9][0-9]{9}$)|(^[0][6-9][0-9]{9}$)|[+][9][1][6-9][0-9]{9}$   ---> Validating Phone number(India)(start with 9876 followed by)

- ^[0-9a-z][a-z0-9_.]{4,13}[a-z0-9][@][0-9a-z]{3,18}[.][a-z]{2,4}$     ---> Email Validation(username@domain.extension
- ^[0-9a-z][0-9a-z_.]{4,13}[0-9a-z]$   - username
        - Length of username : [6 , 15]
        - No special character other than _ and .
        - Should not begin and end with _ and .
        - Character Set : all digits and lower case alphabet _ and .
     - domain
         - Length of domain : [3, 18]
         - No Special characters
         - Character Set : all digits and alphabet
    - extension
        - Length of extention : [2, 4]  (india is .in , .com , .info )
        - No special characters
        - Character Set : alphabet

- ^[a]...[z]$ -> Any string of length 5 that starts with 'a' and ends with 'z'


- ^[a].*[z]$  --> Any string of any length starting with 'a' and ending with 'z'
```

In [ ]:

```

```

In [65]:

```python
# Function to validate a phone number in python
import re      # re is a regular expression

def phoneNumberValidator(number):
    pattern = '(^[6-9][0-9]{9}$)|(^[0][6-9][0-9]{9}$)|[+][9][1][6-9][0-9]{9}
    if re.match(pattern, str(number)):
        return True
    return False
phoneNumberValidator(7997753627)

def emailValidation(email):
    pattern = " ^[0-9a-z][a-z0-9_.]{4,13}[a-z0-9][@][0-9a-z]{3,18}[.][a-z]{2
    if re.match(pattern, email):
        return True
    return False
emailValidation("sireesha7997@gmail.com")

```

Out[65]:

```
False
```

In [19]:
```python
contacts = {}

def addContact(name,phone):
    # verify that the caontact already exit in contacts
    if name not in contacts:
        contacts[name] = phone
        print("Contact %s added"  % name)
    else:
        print("Contact %s already exits"  % name)
    return

addContact("name1","1234567890")
#addContact()

def searchContacts(name):
    if name in contacts:
        print(name, ":", contacts[name])
    else:
        print("%s does not exists" % name)
    return
searchContacts("name1")

# New contacts is given as a dictionary
# Merge new contacts with existing contacts
def importContacts(newContacts):
    contacts.update(newContacts)
    print(len(newContacts.keys())," contacts added successfully")
    return
newContacts = {"name2":9876543210,"name3":6537837637}

importContacts(newContacts)

```

```
Contact name1 added
name1 : 1234567890
2  contacts added successfully
```

In [24]:
```python
contacts = {"name1":[7997753627, 'name1@domain.ext'], "name2":[7868787654, '

def addContact(name,phone):
    # verify that the caontact already exit in contacts
    if name not in contacts and phoneNumberValidator(phone):
        contacts[name] = phone
        print("Contact %s added"  % name)
    if not phoneNumberValidator(phone):
        print("phone number is invalid")
    return True

addContact("name1","9234567890")
```

Out[24]:
```
True
```

In [30]:
```
contacts = {"name1":[7997753627, 'name1@domain.ext'], "name2":[7868787654, '

def addContact(name,phone,email):
    # verify that the caontact already exit in contacts
    if name in contacts:
        print("Name already exists")
    else:
        if phoneNumberValidator(phone):
            print("Invalid Phone Number")
            return
        if not emailValidation(email):
            print("Invalid Email address")
            return
        newContact = []
        newContact.append(phone)
        newContact.append(email)
        contacts[name] = newContact
    return True

addContact("name3",799727,"siri3s@gmailcom")
```

```
Invalid Email address
```

In [32]:
```
def searchContacts(name):
    if name in contacts:
        print(name)
        print("Phone :",contacts[name][0])
        print("Email :", contacts[name][1])
    else:
        print("%s does not exists" % name)
    return
searchContacts("name1")
```

```
name1
Phone : 7997753627
Email : name1@domain.ext
```

In [34]:
```
# New contacts is given as a dictionary
# Merge new contacts with existing contacts
def importContacts(newContacts):
    contacts.update(newContacts)
    print(len(newContacts.keys())," contacts added successfully")
    return
newContacts = {"name4":[9876543210,"name41@domain.ext"],"name5":[6537837637,

importContacts(newContacts)
#contacts
contacts.items()
```

```
2  contacts added successfully
```

Out[34]:
```
dict_items([('name1', [7997753627, 'name1@domain.ext']), ('name2', [7868787654,
'name2@domain.ext']), ('name4', [9876543210, 'name41@domain.ext']), ('name5',
[6537837637, 'name5@domain.ext'])])
```

In [35]:
```python
# Function to list all contacts

def listAllContacts():
    for contact, info in contacts.items():      # info is values
        print(contact, "\n", "Phone :", info[0], "\n", "Email :", info[1])
    return
listAllContacts()
```

```
name1
 Phone : 7997753627
 Email : name1@domain.ext
name2
 Phone : 7868787654
 Email : name2@domain.ext
name4
 Phone : 9876543210
 Email : name41@domain.ext
name5
 Phone : 6537837637
 Email : name5@domain.ext
```

In [ ]:
```python
#  Function to edit (Modify) contact information

def editContact(name, phone, email):

```

```
- We came to file handling because whenever we are stored a appication like contacts in programming they will gone if our program is reseted or permenatly gone...but in file we can store them  permenentaly
```

### ### File Handling in Python

```

File - Document containing information residing
Types - Text, PDF, CSV etc

File I/O - Channelling  I/O data to files
Default I/O channels - Keyboard / Screen

Change I/O channel to files for Reading and writing into files

Read a file - Input from a file
Write to a file - Output to a file

Read / write a file - open(filename, mode)
```

In [40]:
```python
# Function to read a file

def readFile(filename):
    f = open(filename, 'r')
    filedata = f.read()
    f.close()
    return filedata
filename = 'DataFiles/data.txt'
#filedata = readFilele(filename)
# readFile(filename).split('\n')
for line in readFile(filename).split('\n'):
#for line in filedata.split('\n'):
    print(line)
```

```
Line1
Line2
Line3
sireesha
```

In [48]:
```python
def readFile(filename):
    f = open(filename, 'r')
    filedata = f.read()
    f.close()
    return filedata
filename = 'DataFiles/data.txt'
filedata = readFile(filename)
#readFile(filename).split('\n')
#for line in readFile(filename).split('\n'):
for line in filedata.split('\n'):
    print(line)
```

```
Line1
Line2
Line3
sireesha
```

In [50]:
```
def readFile(filename):
    f = open(filename, 'r')
    filedata = f.read()
    f.close()
    return filedata
filename = 'DataFiles/data.txt'
filedata = readFile(filename)
#readFile(filename).split('\n')
#for line in readFile(filename).split('\n'):
#for line in filedata.split('\n'):
   # print(line)


def printFileDataLines(filename):
    f = open(filename, 'r')
    for line in f:
        print(line)
    return
printFileDataLines(filename)
print(readFile(filename))
```

```
Line1

Line2

Line3

sireesha
Line1
Line2
Line3
sireesha
```

In [56]:
```
# Function to write data into a file

def writeIntoFile(filename, filedata):
    with open(filename, 'w') as f:
        f.write(filedata)
    return

filename = 'DataFiles/data.txt'

writeIntoFile(filename, "new data\n")



```

In [64]:
```
# Function to append data to a file

def appendDataToFile(filename, filedata):
    with open(filename, 'a') as f:
        for line in filedata:
            f.write('\n'+line)
    return

filename = 'DataFiles/data.txt'
filedata = ["Line4","Line5"]

appendDataToFile(filename,filedata)
A","E","I","O","U","Y"

```

In [68]:
```
n=input()
if len(str(n))==9:
    for i in range(1,10):
        if((int(n[0])+int(n[1]))%2==0 and (n[2]!='A' or n[2]!='E' or n[2]!='
```

```
12d34-4522
Invalid
```

In [ ]:
```

```