**Exam Problems:**

In [3]:

```python
n = int(input())
s = 0
for i in range(n):
    l = input().split()
    for j in l:
        s+=int(int(j)*-1)
        print(s)



```

```
2
5 -9 12 -10
-5
4
-8
2
4 -3 5 -6
-2
1
-4
2
```

Type *Markdown* and LaTeX: $\alpha^2$

In [6]:

```python
# Pandigital:
n = int(input())
for i in range(n):
    num=input()
    li=[]
    for j in num:
        k = int(j)
        if k not in li:
            li.append(k)
    if len(li)==10:
        print("True")
    else:
        print("False")

```

```
1
0987656454321
True
```

In [7]:

```python
## Another way using lists
#Pan digital means numbers 0-9 contains atleast once in the given input in t
# else return false


n = int(input())
for i in range(n):
    num=input()
    num=set(num)
    if len(num)==10:
        print(True)
    else:
        print(False)

```

```
1
098765443212
True
```

Type *Markdown* and LaTeX: $\alpha^2$

In [9]:

```python
#Alphabet encryption
# Input is
# Hai               -------   Ibj
# Hello how are you -------    Ifkkp ipx bsf zpv


n = int(input())
for i in range(n):
    s = input()
    line= ""
    for j in s:
        if (ord(j)<ord('z') and ord(j) >= ord('a')) or (ord(j)<ord('Z') and
            line += chr(ord(j)+1)
        else:
            line +=j
    print(line)
```

```
1
Hai
Ibj
```

# Sets:

- using sets we can get the unique values in the list i.e set

In [10]:

```python
l = [1,2,1,2,1,2,32455,21455,1,2,23,12]
l =set(l)
l

```

Out[10]:

```
{1, 2, 12, 23, 21455, 32455}
```

In [11]:

```
dir(set)
```

Out[11]:

```
['__and__',
 '__class__',
 '__contains__',
 '__delattr__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__gt__',
 '__hash__',
 '__iand__',
 '__init__',
 '__init_subclass__',
 '__ior__',
 '__isub__',
 '__iter__',
 '__ixor__',
 '__le__',
 '__len__',
 '__lt__',
 '__ne__',
 '__new__',
 '__or__',
 '__rand__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__ror__',
 '__rsub__',
 '__rxor__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__sub__',
 '__subclasshook__',
 '__xor__',
 'add',
 'clear',
 'copy',
 'difference',
 'difference_update',
 'discard',
 'intersection',
 'intersection_update',
 'isdisjoint',
 'issubset',
 'issuperset',
 'pop',
 'remove',
 'symmetric_difference',
 'symmetric_difference_update',
 'union',
 'update']
```

In [16]:

```
l={1,2,3,4,5,6,7,8,1,2,3,4}
m={20,30,40,50,1,2,3,4}
print(m.difference(l))  #it returns a list which removes the common elements
```

```
{40, 50, 20, 30}
```

In [13]:

```
l
```

Out[13]:

```
{1, 2, 3, 4, 5, 6, 7, 8}
```

In [14]:

```
m
```

Out[14]:

```
{1, 2, 3, 4, 20, 30, 40, 50}
```

In [15]:

```
l.difference_update(m) # here we are updating l
l
```

Out[15]:

```
{5, 6, 7, 8}
```

In [19]:

```
l.discard(8)
l
```

Out[19]:

```
{1, 2, 3, 4, 5, 6, 7}
```

In [20]:

```
l.intersection(m)
```

Out[20]:

```
{1, 2, 3, 4}
```

In [22]:

```
l={1,2,3,4,5,6,7,8,9}
l.intersection(m) #Nothing but A intersection b
```

Out[22]:

```
{1, 2, 3, 4}
```

In [23]:

```
l.union(m)  #Nothing but union of both sets but they are unique
```

Out[23]:

```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 20, 30, 40, 50}
```

In [24]:

```
l.add(50)
l
```

Out[24]:

```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 50}
```

In [26]:

```
a = {'Arrow','Apple','Aeroplane'}   #set by default sorts the elements
a
```

Out[26]:

```
{'Aeroplane', 'Apple', 'Arrow'}
```

In [27]:

```
l = {1,2,3,4}
m = {1,2,20,30}
l.intersection(m)
```

Out[27]:

```
{1, 2}
```

In [29]:
```
l.intersection_update(m)
l
```
Out[29]:
```
{1, 2}
```

In [30]:
```
s = 'sravya'
s=set(s)
s

```
Out[30]:
```
{'a', 'r', 's', 'v', 'y'}
```

In [32]:
```
l={1,2,3}
m={4,5,6}
l.isdisjoint(m)
```
Out[32]:
```
True
```

In [34]:
```
l.issubset(m)
```
Out[34]:
```
False
```

In [35]:
```
l={1,2,3,4,5,6}
m={1,2}
m.issubset(l)
```
Out[35]:
```
True
```

In [36]:
```
l.pop() #it removes an element from the front
l
```
Out[36]:
```
{2, 3, 4, 5, 6}
```

In [37]:
```
l.remove(5)  #it removes paticular element which is specified
l
```
Out[37]:
```
{2, 3, 4, 6}
```

In [38]:
```
l={1,2,3,4,5,6}
m={5,6,7,8,9,10,11}
l.symmetric_difference(m)   # (a union b) - (a intersection b)
```
Out[38]:
```
{1, 2, 3, 4, 7, 8, 9, 10, 11}
```

In [39]:
```
l.symmetric_difference_update(m)  #it contains uncommon elements from both t
l
```
Out[39]:
```
{1, 2, 3, 4, 7, 8, 9, 10, 11}
```

In [40]:
```
l.update(m)    #it adds m elements to l set, # you can pass set() values to
l
```
Out[40]:
```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11}
```

Type *Markdown* and LaTeX: $\alpha^2$

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```