In [ ]:
```
a=int(input())
n=list(map(int,input().split()[:a]))
s=0
for i in n:
    s=s+int(i[len(i)//2])
print(s)
```

```
#Data Analysis:
It is a meaningful information said to be Data.
Types of data in realtime/devices
.txt,.doc,.ipnyb,.xls,.ppt,.mp3,.mp4 etc;
Types of Data in realtime
Quantitative
represnets the size/volume of data.
Discrete and Continous
This type of data is Fixed/Constant
Ex: numbers on Dies: varies from 1 to 6
Distance,Color,Nationality,Size etc...
Continous Data means the value/data is Continoues
Height,weight & hours
Qualitative
Data is observed and placed in terms of categories
Ex: Data sunmission in Feedback form
Data in Stastics:
Nominal,Ordinak,Interval and Ratio
Nominal means data is categorized on the basics of names
Ex: nationality,Color,Gender
Ordinal means data is based upon the order of values
Ex: feedback Form
Interval means a range of data values.
Time interval in clock : 2:15 is between 2&3
Ratio means data is meaningfully added,subracted,multiplicated and divided
(ratios)..

Importing modules
Numpy
Pandas
Matplotlib and Seaborn
```

In [1]:
```
import numpy as np
dir(np)
```

```
 'argsort',
 'argwhere',
 'around',
 'array',
 'array2string',

 'array_equal',
 'array_equiv',
 'array_repr',
 'array_split',
 'array_str',
 'asanyarray',
 'asarray',
 'asarray_chkfinite',
 'ascontiguousarray',
 'asfarray',
 'asfortranarray',
 'asmatrix',
 'asscalar',
 'atleast 1d',
```

In [2]:
```
np.__version__
```

Out[2]:
```
'1.20.1'
```

#Arraycreation() -array() is the submodule in numpy module -np.array(iterable)

In [3]:
```
print(np.array([3,4,5,6,9,10]))
```

```
[ 3  4  5  6  9 10]
```

In [4]:
```
#tuple into array
t=np.array((5,6,7,13,90,98))
print(t)
```

```
[ 5  6  7 13 90 98]
```

In [6]:
```
#declaration
ar=np.array([67.9,45.8,56.33,12,78.6],dtype="int")
print(ar)
```

```
[67 45 56 12 78]
```

In [7]:
```
ar=np.array([67.9,45.8,56.33,12,78.6],dtype="float")
print(ar)
```

```
[67.9  45.8  56.33 12.   78.6 ]
```

In [8]:
```
ar=np.array([67.9,45.8,56.33,12,78.6],dtype="str")
print(ar)
```

```
['67.9' '45.8' '56.33' '12' '78.6']
```

In [9]:
```
a=np.array(range(10))
a
```

Out[9]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [10]:
```
a=np.array(range(10,50))
a
```

Out[10]:
```
array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26,
       27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43,
       44, 45, 46, 47, 48, 49])
```

In [11]:
```
a=np.array(range(100,56,-5))
a
```

Out[11]:
```
array([100,  95,  90,  85,  80,  75,  70,  65,  60])
```

In [12]:
```
ar.ndim#dimmensions of array
```

Out[12]:
```
1
```

In [13]:
```
#multidimentional
li=np.array([[9,78,65],[5,4,9]])
li
```

Out[13]:
```
array([[ 9, 78, 65],
       [ 5,  4,  9]])
```

In [14]:
```
li=np.array([[9,78,65],[5,4,9]])            #2d array of size(2x3)
print(li)
```

```
[[ 9 78 65]
 [ 5  4  9]]
```

In [15]:
```
li.size             #represents the siiiize
```

Out[15]:
```
6
```

In [16]:
```
li.shape             #(row,column)
```

Out[16]:
```
(2, 3)
```

In [17]:
```
li.dtype                  #datatype of elements
```

Out[17]:
```
dtype('int32')
```

In [18]:
```
li.itemsize            #size of each item
```

Out[18]:
```
4
```

In [19]:
```
d3=np.array([[3,4,5],[5,6,7],[2,3,4]])
d3
```

Out[19]:
```
array([[3, 4, 5],
       [5, 6, 7],
       [2, 3, 4]])
```

In [20]:
```
d3.ndim #no of arrays
```

Out[20]:
```
2
```

In [21]:
```
#we can create upto 32 dimentional arrays
```

In [22]:
```
dd=np.array([3,4,5,6,7],ndimn=10)
print(dd)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-22-11d81a18a910> in <module>
----> 1 dd=np.array([3,4,5,6,7],ndimn=10)
      2 print(dd)

TypeError: 'ndimn' is an invalid keyword argument for array()
```

In [23]:
```
dd=np.array([3,4,5,6,7],ndmin=10)
print(dd)
```

```
[[[[[[[[[[3 4 5 6 7]]]]]]]]]]
```

In [24]:
```
dd=np.array([3,4,5,6,7],ndimn=33)
print(dd)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-24-97f8f73c2bb5> in <module>
----> 1 dd=np.array([3,4,5,6,7],ndimn=33)
      2 print(dd)

TypeError: 'ndimn' is an invalid keyword argument for array()
```

In [ ]:
```
#creating anoter array
```

In [26]:
```
i=np.eye(5)             #identity matrix
i
```

Out[26]:
```
array([[1., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0.],
       [0., 0., 1., 0., 0.],
       [0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 1.]])
```

In [28]:
```
i2=np.eye(4,5)
i2
```

Out[28]:
```
array([[1., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0.],
       [0., 0., 1., 0., 0.],
       [0., 0., 0., 1., 0.]])
```

In [29]:
```
o=np.ones(5)
o
```

Out[29]:
```
array([1., 1., 1., 1., 1.])
```

In [32]:
```
o=np.ones((4,3))
o
```

Out[32]:
```
array([[1., 1., 1.],
       [1., 1., 1.],
       [1., 1., 1.],
       [1., 1., 1.]])
```

In [33]:
```
f=np.full(4,'hi')          #row size,ele
f
```

Out[33]:
```
array(['hi', 'hi', 'hi', 'hi'], dtype='<U2')
```

In [34]:
```
f=np.full((3,3),'hi')          #row size,ele
f
```

Out[34]:
```
array([['hi', 'hi', 'hi'],
       ['hi', 'hi', 'hi'],
       ['hi', 'hi', 'hi']], dtype='<U2')
```

In [35]:
```
#fill
f.fill(4)
f
```

Out[35]:
```
array([['4', '4', '4'],
       ['4', '4', '4'],
       ['4', '4', '4']], dtype='<U2')
```

In [36]:
```
f.fill(4)
f.dtype='int'
```

In [37]:
```
f
```

Out[37]:
```
array([[52,  0, 52,  0, 52,  0],
       [52,  0, 52,  0, 52,  0],
       [52,  0, 52,  0, 52,  0]])
```

In [38]:
```
#using linspace
ln=np.linspace(10,200,13)
ln
```

Out[38]:
```
array([ 10.        ,  25.83333333,  41.66666667,  57.5       ,
        73.33333333,  89.16666667, 105.        , 120.83333333,
       136.66666667, 152.5       , 168.33333333, 184.16666667,
       200.        ])
```

creating array with arrange()

In [39]:
```
ar=np.arange(25)
ar
```

Out[39]:
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
       17, 18, 19, 20, 21, 22, 23, 24])
```

In [40]:
```
print(np.arange(20,100,7))
```

```
[20 27 34 41 48 55 62 69 76 83 90 97]
```

In [41]:
```
np.arange(100,45,-5)                              #in reverse order
```

Out[41]:
```
array([100,  95,  90,  85,  80,  75,  70,  65,  60,  55,  50])
```

In [43]:
```
ar=ar.reshape(3,4)
print(ar)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-43-4ec81bbde17c> in <module>
----> 1 ar=ar.reshape(3,4)
      2 print(ar)

ValueError: cannot reshape array of size 25 into shape (3,4)
```

In [44]:
```
ar=ar.reshape(5,5)
print(ar)
```

```
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]
 [15 16 17 18 19]
 [20 21 22 23 24]]
```

In [45]:
```
arr=np.arange(100,30,4)
arr
```

Out[45]:
```
array([], dtype=int32)
```

In [46]:
```
ar=np.arange(100,30,4)
ar
```

Out[46]:
```
array([], dtype=int32)
```

In [47]:
```
r2=np.arange(100,20,-4).reshape(4,5)
r2
```

Out[47]:
```
array([[100,  96,  92,  88,  84],
       [ 80,  76,  72,  68,  64],
       [ 60,  56,  52,  48,  44],
       [ 40,  36,  32,  28,  24]])
```

Random

used to generate some random integers in a range
randon.randint()

In [4]:
```
import numpy as np
rn=np.random.randint(10)#0 to 10
rn
```

Out[4]:
```
5
```

In [5]:
```
rn=np.random.randint(10)#0 to 10
rn
```

Out[5]:
```
5
```

In [6]:
```
rn=np.random.randint(10,300)#0 to 10
rn
```

Out[6]:
```
68
```

In [ ]:
```
lw,up=int(input()),int(input())
ri=np.random.randint(lw,up,8)
ri.reshape(2,4)
```

In [12]:
```
lw,up=int(input()),int(input())
ri=np.random.randint(lw,up,8)
ri.reshape(2,4)
```

```
2
11
```

Out[12]:
```
array([[9, 3, 9, 6],
       [6, 5, 3, 4]])
```

In [13]:
```
lw,up=int(input()),int(input())
ri=np.random.randint(lw,up,8)
ri.reshape(2,4)
```

```
100
600
```

Out[13]:
```
array([[174, 335, 121, 243],
       [423, 293, 353, 324]])
```

In [14]:
```
np.random.random((2,5))          #random iint b/w 0 and 1 of size(2,3)
```

Out[14]:
```
array([[0.22461174, 0.97451715, 0.68852476, 0.18689073, 0.64112793],
       [0.29530722, 0.7303158 , 0.22865433, 0.06242181, 0.07513831]])
```

In [15]:
```
np.random.random((2,3))
```

Out[15]:
```
array([[0.08426628, 0.89654281, 0.53218301],
       [0.36177355, 0.8823877 , 0.92634263]])
```

In [16]:
```
np.random.rand(4,9)
```

Out[16]:
```
array([[0.11798399, 0.67327577, 0.84317642, 0.84361321, 0.85825599,
        0.13663027, 0.50078116, 0.12146511, 0.84316819],
       [0.86960968, 0.18989473, 0.89021434, 0.42639796, 0.44276575,
        0.21951427, 0.843481  , 0.19940774, 0.77227808],
       [0.71546939, 0.39991912, 0.81053623, 0.91523743, 0.94322163,
        0.08935305, 0.04735432, 0.15094863, 0.90542747],
       [0.66870656, 0.72988643, 0.65005974, 0.93924331, 0.21865183,
        0.01867074, 0.1613927 , 0.4715179 , 0.27553647]])
```

In [17]:
```
np.random.randn(4,3)  #size
```

...

In [18]:
```
np.linspace(4,10)      #50 partitions
```

Out[18]:
```
array([ 4.        ,  4.12244898,  4.24489796,  4.36734694,  4.48979592,
        4.6122449 ,  4.73469388,  4.85714286,  4.97959184,  5.10204082,
        5.2244898 ,  5.34693878,  5.46938776,  5.59183673,  5.71428571,
        5.83673469,  5.95918367,  6.08163265,  6.20408163,  6.32653061,
        6.44897959,  6.57142857,  6.69387755,  6.81632653,  6.93877551,
        7.06122449,  7.18367347,  7.30612245,  7.42857143,  7.55102041,
        7.67346939,  7.79591837,  7.91836735,  8.04081633,  8.16326531,
        8.28571429,  8.40816327,  8.53061224,  8.65306122,  8.7755102 ,
        8.89795918,  9.02040816,  9.14285714,  9.26530612,  9.3877551 ,
        9.51020408,  9.63265306,  9.75510204,  9.87755102, 10.        ])
```

Accessing the array elemets

- using index
- inddex is of 3 types
- +ve indexing and -ve indexing and fancy indexing
  - +ve index:travrse from left to right
  - starts from 0 to goes to len(it)-1
  - -ve ndexing :from right to left
  - starts from -1 and upto infinite
  - fancy:condition based index

In [19]:
```
r=np.random.randint(50,200,25).reshape(5,-1)
r
```

Out[19]:
```
array([[119, 155, 141, 123,  74],
       [ 60, 109,  54, 136, 193],
       [168, 198,  63, 144, 146],
       [ 93, 193,  82, 152, 112],
       [142, 134, 147, 138,  74]])
```

In [22]:
```
arn=np.arange(50,100,2).reshape(-3,5)
arn
```

Out[22]:
```
array([[50, 52, 54, 56, 58],
       [60, 62, 64, 66, 68],
       [70, 72, 74, 76, 78],
       [80, 82, 84, 86, 88],
       [90, 92, 94, 96, 98]])
```

In [23]:
```
#acessing array elements
arn[3]
```

Out[23]:
```
array([80, 82, 84, 86, 88])
```

In [24]:
```
arn[0]
```

Out[24]:
```
array([50, 52, 54, 56, 58])
```

In [25]:
```
arn[::]
```

Out[25]:
```
array([[50, 52, 54, 56, 58],
       [60, 62, 64, 66, 68],
       [70, 72, 74, 76, 78],
       [80, 82, 84, 86, 88],
       [90, 92, 94, 96, 98]])
```

In [26]:
```
arn[::-1]
```

Out[26]:
```
array([[90, 92, 94, 96, 98],
       [80, 82, 84, 86, 88],
       [70, 72, 74, 76, 78],
       [60, 62, 64, 66, 68],
       [50, 52, 54, 56, 58]])
```

In [27]:
```
arn[::-2]
```

Out[27]:
```
array([[90, 92, 94, 96, 98],
       [70, 72, 74, 76, 78],
       [50, 52, 54, 56, 58]])
```

In [28]:
```
arn[:,1:4]
```

Out[28]:
```
array([[52, 54, 56],
       [62, 64, 66],
       [72, 74, 76],
       [82, 84, 86],
       [92, 94, 96]])
```

In [29]:
```
r
```

Out[29]:
```
array([[119, 155, 141, 123,  74],
       [ 60, 109,  54, 136, 193],
       [168, 198,  63, 144, 146],
       [ 93, 193,  82, 152, 112],
       [142, 134, 147, 138,  74]])
```

In [30]:
```
r[:,1:4]
```

Out[30]:
```
array([[155, 141, 123],
       [109,  54, 136],
       [198,  63, 144],
       [193,  82, 152],
       [134, 147, 138]])
```

In [31]:
```
sub=r[:,1:4]
for line in sub:
    for val in line:
        if val%2==1:
            print(val)
```

```
155
141
123
109
63
193
147
```

In [32]:
```
arn[::2,::2]         #alternating rows and columns
```

Out[32]:
```
array([[50, 54, 58],
       [70, 74, 78],
       [90, 94, 98]])
```

In [33]:
```
arn.transpose()
```

Out[33]:
```
array([[50, 60, 70, 80, 90],
       [52, 62, 72, 82, 92],
       [54, 64, 74, 84, 94],
       [56, 66, 76, 86, 96],
       [58, 68, 78, 88, 98]])
```

In [34]:
```
#fancy index
r<100           #we will get boolean array
```

Out[34]:
```
array([[False, False, False, False,  True],
       [ True, False,  True, False, False],
       [False, False,  True, False, False],
       [ True, False,  True, False, False],
       [False, False, False, False,  True]])
```

In [35]:
```
r[r<100]        #we will get the array
```

Out[35]:
```
array([74, 60, 54, 63, 93, 82, 74])
```

In [36]:
```
#scientific computatons on array
```

In [37]:
```
#to perform all arthemetic operartions on arrays
```

In [38]:
```
print(r+arn)
```

```
[[169 207 195 179 132]
 [120 171 118 202 261]
 [238 270 137 220 224]
 [173 275 166 238 200]
 [232 226 241 234 172]]
```

In [39]:
```
r*arn
```

Out[39]:
```
array([[ 5950,  8060,  7614,  6888,  4292],
       [ 3600,  6758,  3456,  8976, 13124],
       [11760, 14256,  4662, 10944, 11388],
       [ 7440, 15826,  6888, 13072,  9856],
       [12780, 12328, 13818, 13248,  7252]])
```

In [40]:
```
r.arn
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-40-2e18f1abbf54> in <module>
----> 1 r.arn

AttributeError: 'numpy.ndarray' object has no attribute 'arn'
```

In [41]:
```
r.dot(arn)
```

Out[41]:
```
array([[41620, 42844, 44068, 45292, 46516],
       [41570, 42674, 43778, 44882, 45986],
       [49350, 50788, 52226, 53664, 55102],
       [44210, 45474, 46738, 48002, 49266],
       [43130, 44400, 45670, 46940, 48210]])
```

In [42]:
```
print(sum(arn))
r.min()
```

```
[350 360 370 380 390]
```

Out[42]:
```
54
```

In [43]:
```
r.max()
```

Out[43]:
```
198
```

In [44]:
```
r.mean()
```

Out[44]:
```
126.0
```

In [45]:
```
r.median()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-45-6652d53b73d6> in <module>
----> 1 r.median()

AttributeError: 'numpy.ndarray' object has no attribute 'median'
```

In [46]:
```
print(r-arn)
```

```
[[ 69 103  87  67  16]
 [  0  47 -10  70 125]
 [ 98 126 -11  68  68]
 [ 13 111  -2  66  24]
 [ 52  42  53  42 -24]]
```

In [47]:
```
print(r.mean())
```

```
126.0
```

In [48]:
```
r.sum()
```

Out[48]:
```
3150
```

In [49]:
```
arn.sum()
```

Out[49]:
```
1850
```

In [50]:
```
arn[2][2]
```

Out[50]:
```
74
```

In [51]:
```
arn[:][0].sum()
```

Out[51]:
```
270
```

In [52]:
```
#scientific computation
#logarithms and exponentials
```

In [53]:
```
np.log(10)
```

Out[53]:
```
2.302585092994046
```

In [54]:
```
np.log(1)
```

Out[54]:
```
0.0
```

In [55]:
```
np.log(0)
```

```
<ipython-input-55-f6e7c0610b57>:1: RuntimeWarning: divide by zero encountered i
n log
  np.log(0)
```

Out[55]: -inf

In [56]:
```
np.log2(2)
```

Out[56]: 1.0

In [58]:
```
np.log([5,6,7,2])
```

Out[58]: array([1.60943791, 1.79175947, 1.94591015, 0.69314718])

In [61]:
```
np.exp(np.log(1))
```

Out[61]: 1.0

vectorized functions in numpy

In [73]:
```
def greater(a,b):
    if a>b:
        return a
    else:return b
greater(10,3)
```

Out[73]: 10

In [67]:
```
greater([4,5,10],[9,3,1])
```

...

In [68]:
```
li,li2=[[9,3,1],[4,10,4]]
new=[]
for a,b in zip(li,li2):
    if(a>b):
        new.append(a)
    elif b>a:
        new.append(b)
new
```

Out[68]: [9, 10, 4]

In [75]:
```
g=np.vectorize(greater)
g(li,li2)
```

Out[75]: array([9, 10, 4])

In [ ]: