Greedy ATM

```
Available denominations in ATM: 1, 5, 10, 50, 100, 200, 500, 1000, 2000
Amount to be withdrawn:         758
What would be the no of coins/notes dispensed for the given amount, if the ATM
always wants to dispense minimum no of coins/notes so as to save energy?
```

```
1 - 3
5 - 1
50 - 1
200 - 1
500 - 1

Total coins/notes = 7
```

In [13]:
```
l = [1, 5, 10, 50, 100, 200, 500, 1000, 2000]
l.sort(reverse = True)
```

In [14]:
```
print(l)
```

```
[2000, 1000, 500, 200, 100, 50, 10, 5, 1]
```

In [15]:
```
amt = int(input('Enter the amount to withdraw: '))
print(amt)
```

```
Enter the amount to withdraw: 758
758
```

In [16]:
```
coins = 0
for val in l:
    noc = amt // val # noc = 758 // 500
    amt = amt - (noc * val)
    coins += noc

print('No of coins:', coins)
```

```
No of coins: 7
```

In [17]:
```
l = [7, 6, 10, 1, 3]
amt = 9
l.sort(reverse = True)
print(l)
```

```
[10, 7, 6, 3, 1]
```

In [18]:
```
coins = 0
for val in l:
    noc = amt // val # noc = 2 // 1 = 2
    amt = amt - (noc * val) # amt = 2 - (2 * 1) = 0
    coins += noc

print('No of coins:', coins)
```

```
No of coins: 3
```

Array Rotation

```
li = [9, 6, 5, 0, 8, 2]
What would be the array content after rotating the array towards left 'k'
times. Solve in O(n).
```

```
9, 6, 5, 0, 8, 2
6, 5, 0, 8, 2, 9 - 1st rotation
5, 0, 8, 2, 9, 6 - 2nd rotation
0, 8, 2, 9, 6, 5 - 3rd rotation

Time Cpomplexity for one circular left rotation: O(n)
k rotations = O(k * n)
if k = n
T(n) = O(n^2)
```

```
li = [9, 6, 5, 0, 8, 2]

0, 8, 2, 9, 6, 5
6 left rotations = same array (there are 6 elements)
k = 9 ~~ rotating 3 times (9 % 6)
effective k = k % n
```

In [21]:
```
li = [9, 6, 5, 0, 8, 2]
k = int(input('Enter no of rotations: '))
k = k % len(li)

print(li[k : ] + li[0 : k]) # O(n)
```

```
Enter no of rotations: 2
[5, 0, 8, 2, 9, 6]
```

In [ ]: