```python
def leetcoder(leetText):
    d={"0":"O","2":"Z","3":"E","7":"L","#":"H","@":"A","$":"S"}
    res=[]
    for word in leetText.split():
        if any(c.isalpha() for c in word) and any(c in d for c in word):
            word="".join(d.get(c,c) for c in word)
        res.append(word.upper())
    return " ".join(res)
```

```python
def shopping_total(items):
    t=0
    for item in items:
        name,price=item
        if price>50:
            price=price*0.9
        else:
            price=price
        t+=price
    return round(t,1)
```

```python
def decipher(secret,shift):
    res=""
    for ch in secret:
        if 'A'<=ch<='Z':
            new_ch=(ord(ch)-ord('A')-shift)%26+ord('A')
            res+=chr(new_ch)
        elif 'a'<=ch<='z':
            new_ch=(ord(ch)-ord('a')-shift)%26+ord('a')
            res+=chr(new_ch)
        else:
            res+=ch
    return res
```

```python
class Book:
    def __init__(self,name,author,year):
        self.__name=name
        self.__author=author
        self.__year=year
        self.borrowed=False
    def __str__(self):
        return f"Book details: {self.__name} by {self.__author}, published in {self.__year}"
    def borrow_status(self):
        return f"Borrowed: {self.borrowed}"
    def set_name(self,new_name):
        raise AttributeError("Cannot set attribute")
    @property
    def name(self):
        return self.__name
```

```python
def password_strength(password,k):
    if len(password)<k and k<=0:
        return 0
    unique=set()
    for i in range(len(password)-k+1):
        substring=password[i:i+k]
        unique.add(substring)
    return len(unique)
```

```python
def NegativeValueError(ValueError):
    def __init__(self,value):
        self.value=value
        super().__init__("Input value is negative")
def validate_positive(value):
    if value>0:
        return value
    else:
        raise NegativeValueError(value)
```