Use Case : Input problem
ID : st001
Actor : general user
Precondition : -
Flow of Event 1.get problem 2.check syntax problem 3.return problem
Postcondition : value,valueFind,cost

Use Case : Find Equation
ID : fe001
Actor : program
Precondition : value,valueFind,valueOfEquation[]
Flow of Event 1.get value,valueFind,valueOfEquation[] 2 call function callEquation(value,valueFind,valueOfEquation): list a=value+valueFind if a in valueOfEquation[]: return valueOfEquation[valueOfEquation.index(a)] else : a=min(valueOfEquation - (value+valueFind)) call function calEquation(value,valueFind,valueOfEquation): 3 return list index of equation
Postcondition : index of equation

Use Case : Find Equation

ID : fe002

Actor : program

Precondition : value,valueFind,index of equation,cost,equation

Flow of Event

1 get value,valueFind,equation,index

2 call function changEquation(value,valueFind,equation[index]):

```
left=equation[:equation.index('=')-1]
right=equation[equation.index('=')+1:]
if valuefind in right
    swap left,right
if value in left >1
    return call function poly
        return left+rigth+'=0'
give value of left is not valueFind swap to rigth
    return left+'='+right
```

3 get result of changEquation function to equationChanged value

4 call function calValue(equationChanged,value,cost)

```
subE=gorup(equation)
if subE=1:return cost[cost.index(subEquation[0])
if subE=2:return operation.index(subE[0])calValue(subE[1])
else:return calValue(subE[0]) operation.index(subE[1]) calValue(subE[2])
```

5 get result of calValue to result of problem

Postcondition : equationChanged,result of problem

Use Case : Show solving problem	
ID	: fn 001
Actor	: general user
Precondition	: index of equation,equationChanged,result of problem
Flow of Event 1. get index of equation,equationChanged,result of problem 2.show index of equation,equationChanged,result of problem	
Postcondition	: -