Wallet Overloading

locked

Create a `Wallet` class in C++ to store details of money kept in a wallet. You need to maintain information about all possible denominations of money present in the wallet: notes in denomination of 2000, 500, 200, 100, 50, 20, 10, 5, coins in denomination of 20, 10, 5, 2, 1, 0.5, 0.25. You also need to implement the following operations:

- Overload `>>` operator to input wallet details from `cin` (input format is given below).
- Overload `<<` operator to print the wallet details (output format is given below).
- Overload `double` operator to convert the wallet amount to a `double`, representing the wallet balance.
- Overload `[]` operator to return the number of notes/coins present in the wallet in the input denomination. For example, if `w` is a `Wallet`, then `w[50]` should return the number of Rs. 50 notes, while `w[10]` should return the sum of Rs. 10 notes and coins in the wallet.
- Overload `+` operator to add two `Wallet`s. This should simply add up the corresponding denominations in both wallets.
- Overload `+` operator to add a `Wallet w` and a `double d`. Here, the objective is to add `d` amount of money to the wallet `w` by picking the denominations in such a way that the number of notes and coins added to `w` is minimized. For example, if we want to add `545.5`, then your implementation should add 1 Rs. 500 note, 2 Rs. 20 notes, 1 Rs. 5 note and Rs 0.50 coin. Prefer notes over coins. You can assume that `d` can be broken down using the available denominations.
- Overload `-` operator to subtract two `Wallet`s. This should simply subtract the correspondong denominations in the wallets. You can assume that the wallet subtracted from will have sufficient notes/coins.
- Overload `*` operator to multiply two `Wallet`s. Here, the objective is consider the balance amount in the second wallet, multiply it with the balance amount of the first wallet (this becomes the target amount), and then add notes/coins to the first wallet so that its balance reaches the target amount. Again, the total number of notes/coins to be added to the first wallet to reach the target amount should be minimized.
- Overload `*` operator to multiply a `Wallet` and an `double d`. Here, you need to first multiply the wallet balance and the double to get a target amount, and then add notes/coins (while minimizing their total number) to reach that amount. You can assume that `d` > 0.

Input Format

- The input will consist of a sequence of commands. The starter code already handles the different command options.
- Wallet contents will be specified as a space-separated list of integers: $x_{2000}\ x_{500}\ x_{200}\ x_{100}\ x_{50}\ x_{20n}\ x_{10n}\ x_{5n}\ x_{20c}\ x_{10c}\ x_{5c}\ x_2\ x_1\ x_{0.5}\ x_{0.25}$, where x_y indicates the number of notes/coins of denomination y. Note that x_{yn} denotes notes, while x_{yc} denotes coins of denomination y.

Constraints

Number of notes/coins of a denomination in a wallet ≤ 100

Output Format

- Wallet details should be printed in the following format: $(2000\ x_{2000})\ (500\ x_{500}) \ldots (0.25\ x_{0.25})$, where x_y denotes the number of y-denomination notes/coins in the wallet. Use $(yn\ x_{yn})$ and $(yc\ x_{yc})$ to denote notes and coins of denomination y respectively. The ordering should be the same as the input format (see the sample testcases for better understanding).

Sample Input 0

```
2
1 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
2
```

Sample Output 0

```
Wallet contains: (2000 15) (500 14) (200 13) (100 12) (50 11) (20n 10) (10n 9) (5n 8) (20c 7) (10c 6) (5c 5) (2
4) (1 3) (0.5 2) (0.25 1)
```

Sample Input 1

```
3
1 15 14 13 12 11 10 9 8 7 6 5 4 3 2 2
2
3
```

Sample Output 1

```
Wallet contains: (2000 15) (500 14) (200 13) (100 12) (50 11) (20n 10) (10n 9) (5n 8) (20c 7) (10c 6) (5c 5) (2
4) (1 3) (0.5 2) (0.25 2)
Wallet has balance: 41917.5
```

Sample Input 2

```
14
1 1 3 13 12 11 10 9 8 7 6 5 4 3 2 1
4 2000
4 500
4 200
4 100
4 50
4 20
4 10
4 5
4 2
4 1
4 0.5
4 0.25
2
```

Sample Output 2

```
Wallet contains 1 number of notes/coins of denomination 2000
Wallet contains 3 number of notes/coins of denomination 500
Wallet contains 13 number of notes/coins of denomination 200
Wallet contains 12 number of notes/coins of denomination 100
Wallet contains 11 number of notes/coins of denomination 50
Wallet contains 17 number of notes/coins of denomination 20
Wallet contains 15 number of notes/coins of denomination 10
Wallet contains 13 number of notes/coins of denomination 5
Wallet contains 4 number of notes/coins of denomination 2
Wallet contains 3 number of notes/coins of denomination 1
Wallet contains 2 number of notes/coins of denomination 0.5
Wallet contains 1 number of notes/coins of denomination 0.25
Wallet contains: (2000 1) (500 3) (200 13) (100 12) (50 11) (20n 10) (10n 9) (5n 8) (20c 7) (10c 6) (5c 5) (2 4)
(1 3) (0.5 2) (0.25 1)
```

Sample Input 3

```
4
1 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
2
5 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
2
```

Sample Output 3

```
Wallet contains: (2000 15) (500 14) (200 13) (100 12) (50 11) (20n 10) (10n 9) (5n 8) (20c 7) (10c 6) (5c 5) (2
4) (1 3) (0.5 2) (0.25 1)
Wallet contains: (2000 30) (500 28) (200 26) (100 24) (50 22) (20n 20) (10n 18) (5n 16) (20c 14) (10c 12) (5c
10) (2 8) (1 6) (0.5 4) (0.25 2)
```

Sample Input 4

```
4
1 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
2
6 2678.5
2
```

Sample Output 4

```
Wallet contains: (2000 15) (500 14) (200 13) (100 12) (50 11) (20n 10) (10n 9) (5n 8) (20c 7) (10c 6) (5c 5) (2
4) (1 3) (0.5 2) (0.25 1)
Wallet contains: (2000 16) (500 15) (200 13) (100 13) (50 12) (20n 11) (10n 9) (5n 9) (20c 7) (10c 6) (5c 5) (2
5) (1 4) (0.5 3) (0.25 1)
```

Sample Input 5

```
4
1 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
2
7 15 14 13 0 11 10 0 8 7 6 5 0 3 2 1
2
```

Sample Output 5

```
Wallet contains: (2000 15) (500 14) (200 13) (100 12) (50 11) (20n 10) (10n 9) (5n 8) (20c 7) (10c 6) (5c 5) (2
4) (1 3) (0.5 2) (0.25 1)
Wallet contains: (2000 0) (500 0) (200 0) (100 12) (50 0) (20n 0) (10n 9) (5n 0) (20c 0) (10c 0) (5c 0) (2 4) (1
0) (0.5 0) (0.25 0)
```

Sample Input 6

```
5
1 0 0 1 1 0 1 2 0 1 3 1 2 3 2 1
3
8 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0
3
2
```

Sample Output 6

```
Wallet has balance: 403.25
Wallet has balance: 6048.75
Wallet contains: (2000 3) (500 0) (200 0) (100 0) (50 0) (20n 2) (10n 0) (5n 1) (20c 0) (10c 0) (5c 0) (2 1) (1
1) (0.5 1) (0.25 1)
```

Sample Input 7

```
5
1 0 1 0 1 0 1 2 0 1 3 1 2 3 2 1
3
9 5
3
2
```

Sample Output 7

```
Wallet has balance: 703.25
Wallet has balance: 3516.25
Wallet contains: (2000 1) (500 3) (200 0) (100 0) (50 0) (20n 0) (10n 1) (5n 1) (20c 0) (10c 0) (5c 0) (2 0) (1
1) (0.5 0) (0.25 1)
```

Sample Input 8

```
6
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
2
8 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
2
5 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
2
```

Sample Output 8

```
Wallet contains: (2000 2) (500 3) (200 4) (100 5) (50 6) (20n 7) (10n 8) (5n 9) (20c 10) (10c 11) (5c 12) (2 13)
(1 14) (0.5 15) (0.25 16)
Wallet contains: (2000 18931) (500 3) (200 1) (100 1) (50 0) (20n 0) (10n 0) (5n 0) (20c 0) (10c 0) (5c 0) (2 1)
(1 0) (0.5 1) (0.25 1)
Wallet contains: (2000 18932) (500 5) (200 4) (100 5) (50 5) (20n 6) (10n 7) (5n 8) (20c 9) (10c 10) (5c 11) (2
13) (1 13) (0.5 15) (0.25 16)
```

Sample Input 9

```
8
1 2 3 4 5 6 7 8 9 10 11 14 15 22 21 20
4 20
4 1
5 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
7 1 2 3 2 3 1 7 8 9 10 11 12 13 14 15
3
9 5.5
2
```

Sample Output 9

```
Wallet contains 17 number of notes/coins of denomination 20
Wallet contains 22 number of notes/coins of denomination 1
Wallet has balance: 8212.5
Wallet contains: (2000 22) (500 2) (200 0) (100 1) (50 1) (20n 0) (10n 1) (5n 1) (20c 0) (10c 0) (5c 0) (2 1) (1
1) (0.5 1) (0.25 1)
```

Submissions: 97

Max Score: 100
Difficulty: Medium

Rate This Challenge:
☆☆☆☆☆

More

C++20

```cpp
#include <cmath>
#include <cstdio>
#include <vector>
#include <iostream>
#include <algorithm>
using namespace std;

/*Define your Wallet class here*/
class Wallet
{
    private:
       int arr[15];
       double help[15] = {2000, 500, 200, 100, 50, 20, 10, 5, 20, 10, 5, 2, 1, 0.5, 0.25};
       int hii[3];
       double bal;
    public:
    Wallet()
    {
        bal = 0.00;
    }
    int & operator [] (double hello)
    {
        int gg = hello*(100);
        switch (gg)
        {
            case 200000: return arr[0];break;
            case 50000: return arr[1];break;
            case 20000: return arr[2];break;
            case 10000: return arr[3];break;
            case 5000:return arr[4];break;
            case 2000:return hii[0];break;
            case 1000:return hii[1];break;
            case 500:return hii[2];break;
            case 200:return arr[11];break;
            case 100:return arr[12];break;
            case 50:return arr[13];break;
            case 25:return arr[14];break;
            default: break;
        }
        return arr[0];
    }
    friend istream & operator >> (istream & in, Wallet & w);
    friend ostream & operator << (ostream & out, Wallet & w);
    operator double () const { return bal;}
    void operator +(Wallet w)
    {
        for (int ii=0; ii<15; ii++)
        {
            arr[ii]+=w.arr[ii];
        }
        hii[0]=arr[5] + arr[8];
        hii[1]=arr[6] + arr[9];
        hii[2]=arr[7] + arr[10];
        bal+=w.bal;
    }
    void operator -(Wallet w)
    {
        for (int ii=0; ii<15; ii++)
        {
            arr[ii]-=w.arr[ii];
        }
        hii[0]=arr[5] + arr[8];
        hii[1]=arr[6] + arr[9];
        hii[2]=arr[7] + arr[10];
        bal-=w.bal;
    }
    void operator +(double d)
    {
        int brr[15];
        int h = d;
        brr[0] = h/2000;
        h-=brr[0]*2000;
        brr[1] = h/500;
        h-=brr[1]*500;
        brr[2] = h/200;
        h-=brr[2]*200;
        brr[3] = h/100;
        h-=brr[3]*100;
        brr[4] = h/50;
        h-=brr[4]*50;
        brr[5] = h/20;
        h-=brr[5]*20;
        brr[6] = h/10;
        h-=brr[6]*10;
        brr[7] = h/5;
        h-=brr[7]*5;
        brr[8]=0;
        brr[9]=0;
        brr[10]=0;
        brr[11] = h/2;
        h-=brr[11]*2;
        brr[12] = h/1;
        h-=brr[12]*1;
        int kk = d;
        double ll = d - (double)kk;
        ll = ll * 100;
        int yy = ll;
        brr[13] = yy/50;
        yy = yy - brr[13]*50;
        brr[14] = yy/25;
        for (int ii=0; ii<15; ii++)
        {
            arr[ii]+=brr[ii];
        }
        hii[0]=arr[5]+arr[8];
        hii[1]=arr[6]+arr[9];
        hii[2]=arr[7]+arr[10];
        bal+=d;
    }
    void operator *(Wallet w1)
    {
        arr[5]=arr[5] + arr[8];
        arr[6]=arr[6] + arr[9];
        arr[7]=arr[7] + arr[10];
        bal = bal * ((double) w1);
        int h = bal;
        arr[0] = h/2000;
        h-=arr[0]*2000;
        arr[1] = h/500;
        h-=arr[1]*500;
        arr[2] = h/200;
        h-=arr[2]*200;
        arr[3] = h/100;
        h-=arr[3]*100;
        arr[4] = h/50;
        h-=arr[4]*50;
        arr[5] = h/20;
        h-=arr[5]*20;
        arr[6] = h/10;
        h-=arr[6]*10;
        arr[7] = h/5;
        h-=arr[7]*5;
        arr[8]=0;
        arr[9]=0;
        arr[10]=0;
        arr[11] = h/2;
        h-=arr[11]*2;
        arr[12] = h/1;
        h-=arr[12]*1;
        int kk = bal;
        double ll = bal - (double)kk;
        ll = ll * 100;
        int yy = ll;
        arr[13] = yy/50;
        yy = yy - arr[13]*50;
        arr[14] = yy/25;
        hii[0]=arr[5];
        hii[1]=arr[6];
        hii[2]=arr[7];
        bal = kk + arr[13]*0.5 + arr[14]*0.25;
    }
    void operator *(double d)
    {
        arr[5]=arr[5] + arr[8];
        arr[6]=arr[6] + arr[9];
        arr[7]=arr[7] + arr[10];
        bal = bal * d;
        int h = bal;
        arr[0] = h/2000;
        h-=arr[0]*2000;
        arr[1] = h/500;
        h-=arr[1]*500;
        arr[2] = h/200;
        h-=arr[2]*200;
        arr[3] = h/100;
        h-=arr[3]*100;
        arr[4] = h/50;
        h-=arr[4]*50;
        arr[5] = h/20;
        h-=arr[5]*20;
        arr[6] = h/10;
        h-=arr[6]*10;
        arr[7] = h/5;
        h-=arr[7]*5;
        arr[8]=0;
        arr[9]=0;
        arr[10]=0;
        arr[11] = h/2;
        h-=arr[11]*2;
        arr[12] = h/1;
        h-=arr[12]*1;
        int kk = bal;
        double ll = bal - (double)kk;
        ll = ll * 100;
        int yy = ll;
        arr[13] = yy/50;
        yy = yy - arr[13]*50;
        arr[14] = yy/25;
        bal = kk + arr[13]*0.5 + arr[14]*0.25;
        hii[0]=arr[5];
        hii[1]=arr[6];
        hii[2]=arr[7];
    }

};
istream & operator >> (istream & in, Wallet & w)
{
    for (int ii=0; ii<15; ii++)
    {
        int k;
        cin>>k;
        w.arr[ii]=k;
    }
    w.bal=w.arr[0]*2000.00 + w.arr[1]*500.00 + w.arr[2]*200.00 + w.arr[3]*100.00 + w.arr[4]*50.00
+ w.arr[5]*20.00 + w.arr[6]*10.00 + w.arr[7]*5.00 + w.arr[8]*20.00 + w.arr[9]*10.00 +
w.arr[10]*5.00 + w.arr[11]*2.00 + w.arr[12]*1.00 + w.arr[13]*0.50 + w.arr[14]*0.25;
    w.hii[0]=w.arr[5] + w.arr[8];
    w.hii[1]=w.arr[6] + w.arr[9];
    w.hii[2]=w.arr[7] + w.arr[10];


    return in;
}
ostream & operator << (ostream & out, Wallet & w)
{
    return out << "(2000 " << w.arr[0]<<") "<< "(500 "<< w.arr[1] <<") " <<"(200 "<<w.arr[2] <<")
"<< "(100 "<<w.arr[3]<<") "<<"(50 "<<w.arr[4]<<") " << "(20n "<<w.arr[5]<<") "<<"(10n "<<w.arr[6]
<<") "<<"(5n "<<w.arr[7]<<") "<<"(20c "<<w.arr[8]<<") "<< "(10c "<<w.arr[9]<<") "<<"(5c "
<<w.arr[10]<<") "<<"(2 "<<w.arr[11]<<") "<<"(1 "<<w.arr[12]<<") "<<"(0.5 "<<w.arr[13]<<") "<<"
(0.25 "<<w.arr[14]<<")";
}
int main()
{
    int N;
    cin >> N;
    Wallet w,w1;
    int command;
    double d;
    for (int i = 0; i < N; i++)
    {
        cin >> command;
        switch (command)
        {
            case 1: /*initialize wallet*/
                cin >> w;
                break;

            case 2: /*print wallet*/
                cout << "Wallet contains: " << w << endl;
                break;

            case 3: /*wallet balance*/
                cout << "Wallet has balance: " << (double) w << endl;
                break;

            case 4: /*denomination query*/
                cin >> d;
                cout << "Wallet contains " << w[d] << " number of notes/coins of denomination "
<< d << endl;
                break;

            case 5: /*add two wallets*/
                cin >> w1;
                w + w1;
                break;

            case 6: /*add money to wallet*/
                cin >> d;
                w + d;
                break;

            case 7: /*subtract from wallet*/
                cin >> w1;
                w - w1;
                break;

            case 8: /*multiply wallets*/
                cin >> w1;
                w * w1;
                break;

            case 9: /*multiply wallet and double*/
                cin >> d;
                w * d;
                break;

            default:
                break;
        }
    }
}
```

Line: 23 Col: 30

Upload Code as File

Test against custom input

Run Code

Submit Code

Interview Prep | Blog | Scoring | Environment | FAQ | About Us | Support | Careers | Terms Of Service | Privacy Policy |