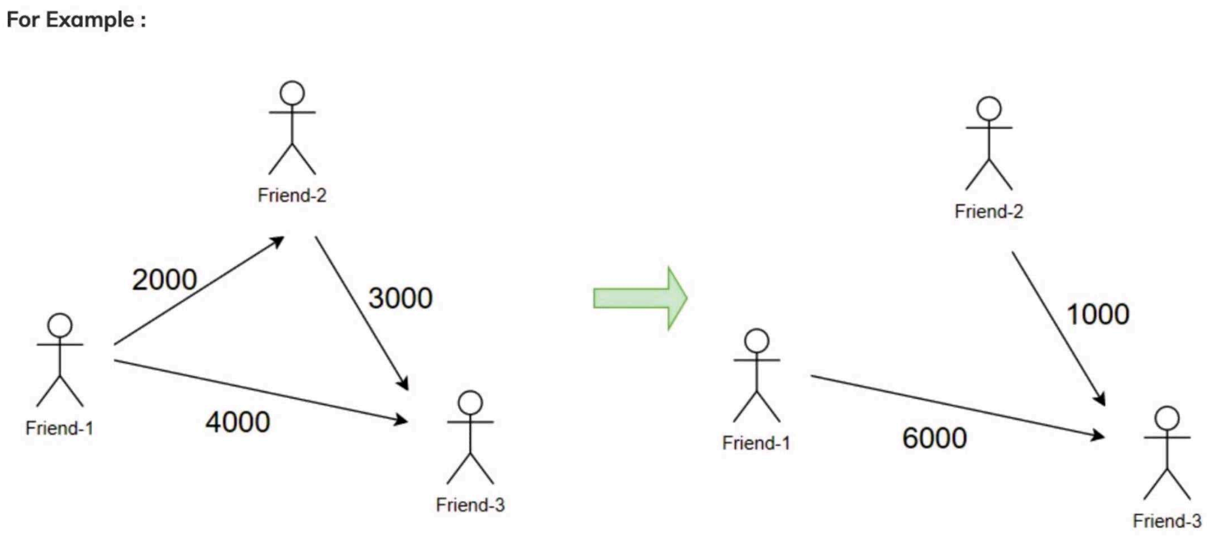
1. **Problem:** Minimizing Cash flow among Friends

Problem Definition:

Here there are N no.of people who have borrowed money from each other. Instead of exchanging money many times we'll be using a greedy approach to make the number of transactions as low as possible.

Problem Explaination:

Once the net amount for every person is evaluated, find two persons with maximum and minimum net amounts. These two persons are the most creditors and debtors. The person with a minimum of two is our first person to be settled and removed from the list. Let the minimum of two amounts be x. We pay 'x' amount from the maximum debtor to maximum creditor and settle one person. If x is equal to the maximum debit, then maximum debtor is settled, else maximum creditor is settled.



In figure 1 : friend-1 has to pay 2000$ to friend-2, and 4000$ to friend-3 and friend-2 has to pay 3000$ to friend-3.

In figure 2 : so we can minimize the flow between friend-1 to friend-2 by direct pay to friend-1 to friend-3

2. GREEDY APPROACH:

The approach we will be using for minimizing cash flow is the Greedy Algorithm. The greedy approach is used to build the solution in pieces, and this is what we want to minimize the cash flow. At every step, we will settle all the amounts of one person and recur for the remaining n-1 persons.
Calculate the net amount for every person, which can be calculated by subtracting all the debts, i.e., the amount to be paid from all credit, i.e., the amount to be paid to him. After this, we will find two persons with the maximum and the minimum net amounts. The person with a minimum of two is our first person to be settled and removed from the list.
Following algorithm will be done for every person varying 'i' from 0 to n-1.

1. The first step will be to calculate the net amount for every person and store it in an amount array.
    1. Net amount = sum(received money) - sum(sent money).
2. Find the two people that have the most credit and the most debt. Let the maximum amount to be credited from maximum creditor be max_credit and the maximum amount to be debited from maximum debtor be max_debit. Let the maximum debtor be debt and maximum creditor be cred.
3. Let the minimum of two amounts be 'y'.
4. If y equals max_credit, delete cred from the list and repeat for the remaining (n-1) people.
5. If y equals max_debit, delete debt from the group of people and repeat for recursion.
6. If the amount is 0, then the settlement is done.

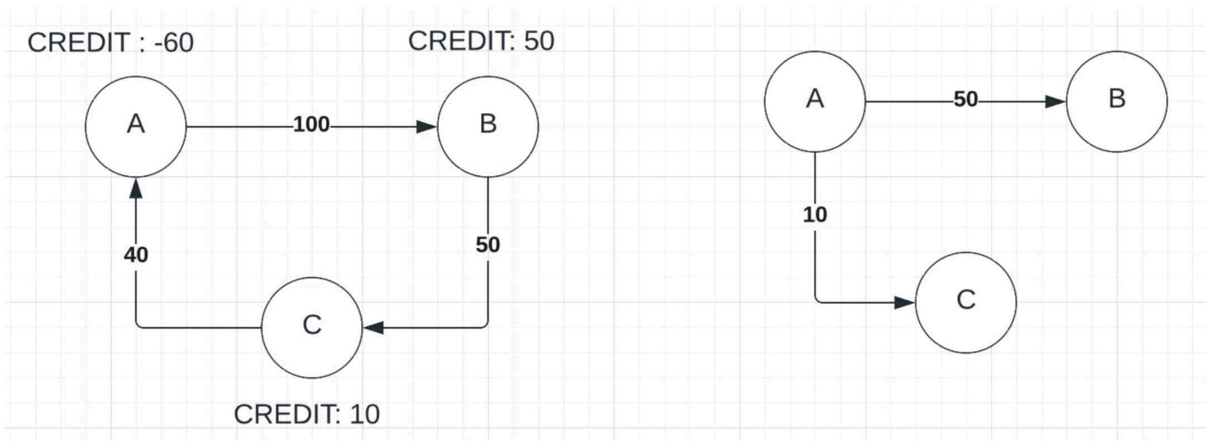
Algorithm:

- Create a net amount array ‘netAmount’
- Fill this array, Now for every friend ‘i’
    - ‘netAmount[i]’= sum of all received money by ‘i-th’ friend - the sum of all sent money by ‘i-th’ friend.
- Create a 2-D matrix to store the ‘answer’.
- Iterate a while loop until all the values of ‘netAmount’ is not ‘0’
    - Find the minimum and maximum of ‘netAmount’
    - Suppose ‘x’ index value is the max net amount and ‘y’ index is min net amount, then:
        - ‘netAmount[x] = netAmount[x]- abs(netAmount[y])’
        - Update ‘answer[y][x] = abs(netAmount[y])
        - It represents that the ‘y-th’ friend will pay ‘netAmount[y]’ to ‘x-th’ friend.
        - Set ‘netAmount[y] = 0’.
- In the end, return ‘answer’.

```cpp
#include<iostream>
using namespace std;
#define N 3
int getMin(int arr[])
{
    int minind = 0;
    for (int i=1; i<N; i++)
        if (arr[i] < arr[minind])
            minind = i;
    return minind;
}

int getMax(int arr[])
{
    int maxind = 0;
    for (int i=1; i<N; i++)
        if (arr[i] > arr[maxind])
            maxind = i;
    return maxind;
}

int minOf2(int x, int y)
{
    return (x<y)?x: y;
}

void minCashFlowRec(int amount[])
{
    int mxCredit = getMax(amount), mxDebit = getMin(amount);

    if (amount[mxCredit] == 0 && amount[mxDebit] == 0)
        return;

    int min = minOf2(-amount[mxDebit], amount[mxCredit]);
    amount[mxCredit] -= min;
    amount[mxDebit] += min;

    cout << "Person " << mxDebit << " pays " << min
        << " to " << "Person " << mxCredit << endl;

    minCashFlowRec(amount);
}
void minCashFlow(int graph[][N])
{
    int amount[N] = {0};

    for (int p=0; p<N; p++)
    for (int i=0; i<N; i++)
        amount[p] += (graph[i][p] - graph[p][i]);

    minCashFlowRec(amount);
}
int main()
{
    int graph[N][N] = { {0, 1000, 2000},
                        {0, 0, 5000},
                        {0, 0, 0},};

    minCashFlow(graph);
    return 0;
}
```

```
Person 1 pays 4000 to Person 2
Person 0 pays 3000 to Person 2
```

3. **Time Complexity analysis:**



So the idea is,

1. We can pick the person who has the largest debit and try to settle the person who has the largest credit. So here if A gives 50 to B then the credit of B will become zero and we'll have 1 transaction as of now.
2. This ensures that A will now have 10 and you can see that A is still in the debit mode and from the credit line next person is C(next largest amount).
3. Try to settle these amounts. So we can send 10 to C and A will now have zero amount and C will also be zero.
4. So givers and takers are now having null balance, hence we have successfully balanced the amount and minimum number of transactions happened here is 1 + 1 i.e 2.

So making a Max Heap for a debit and credit list will solve this problem in **O(nlog(n))** time complexity.

CONCLUSION :

Thus, we had completed the implementation of code using greedy algorithm.

References:

https://www.programiz.com/dsa/greedy-algorithm
https://github.com/topics/greedy-algorithm?o=asc&s=stars