Assignement_3.2_Practice Problem 2 (Split the Bill)

Code Documentation:

```
def solve(N, trans_list):

    bal = {}  # Dictionary to store balances of individuals

    for trans in trans_list: # Iterate over each transaction
        for payer, amt in trans['paid_by']: # Update balances for payments made by individuals
            bal[payer] = bal.get(payer, 0) + amt
        for borrower, amt in trans['split_as']: # Update balances for amounts to be split among individuals
            bal[borrower] = bal.get(borrower, 0) - amt

    payments = []  # List to store payment details

    while True: # Iterate until all debts are settled
        borrowers = sorted(filter(lambda x: bal[x] < 0, bal)) # Sort individuals with negative balances (borrowers)
        lenders = sorted(filter(lambda x: bal[x] > 0, bal)) # Sort individuals with positive balances (lenders)
        if not borrowers or not lenders: # Break loop if no borrowers or lenders left
            break
        borrower, lender = borrowers[0], lenders[0]  # Select borrower and lender
        amt = min(abs(bal[borrower]), bal[lender])  # Determine payment amount
        payments.append([lender, borrower, amt])  # Add payment details

        bal[borrower] += amt  # Update balances after payment
        bal[lender] -= amt

    print(len(payments)) # Print the number of payments needed and the details of each payment
    for payment in payments:
        print(*payment)



def solve(N, trans_list):

    bal = {}  # Dictionary to store balances of individuals

    for trans in trans_list: # Iterate over each transaction
        for payer, amt in trans['paid_by']: # Update balances for payments made by individuals
            bal[payer] = bal.get(payer, 0) + amt
        for borrower, amt in trans['split_as']: # Update balances for amounts to be split among individuals
            bal[borrower] = bal.get(borrower, 0) - amt

    payments = []  # List to store payment details

    while True: # Iterate until all debts are settled
        borrowers = sorted(filter(lambda x: bal[x] < 0, bal)) # Sort individuals with negative balances (borrowers)
        lenders = sorted(filter(lambda x: bal[x] > 0, bal)) # Sort individuals with positive balances (lenders)
        if not borrowers or not lenders: # Break loop if no borrowers or lenders left
            break
        borrower, lender = borrowers[0], lenders[0]  # Select borrower and lender
        amt = min(abs(bal[borrower]), bal[lender])  # Determine payment amount
        payments.append([lender, borrower, amt])  # Add payment details

        bal[borrower] += amt  # Update balances after payment
        bal[lender] -= amt

    print(len(payments)) # Print the number of payments needed and the details of each payment
    for payment in payments:
        print(*payment)
```

Code Demonstration:

```
def solve(N, trans_list):
    bal = {}

    for trans in trans_list:
        for payer, amt in trans['paid_by']:
            bal[payer] = bal.get(payer, 0) + amt
        for borrower, amt in trans['split_as']:
            bal[borrower] = bal.get(borrower, 0) - amt

    payments = []

    while True:
        borrowers = sorted(filter(lambda x: bal[x] < 0, bal))
        lenders = sorted(filter(lambda x: bal[x] > 0, bal))
        if not borrowers or not lenders:
            break

        borrower, lender = borrowers[0], lenders[0]
        amt = min(abs(bal[borrower]), bal[lender])
        payments.append([lender, borrower, amt])

        bal[borrower] += amt
        bal[lender] -= amt

    print(len(payments))
    for payment in payments:
        print(payment[1], payment[0], payment[2])



print("For all data inputs follow this format (1 60) ")
print("Enter the number of people in the group (n) and the number of transactions recorded (m):")

N, M = map(int, input().split())# Read the number of people and transactions

if not (2 <= N <= 2 * 10**5) or not (1 <= M <= 5000): # Check if N and M fall within the specified constraints
    print("Invalid input for N or M. Please ensure 2 <= N <= 2 * 10^5 and 1 <= M <= 5000.")
    exit()

trans_list = []  # List to store transaction details

print("Enter details for each transaction:")

for i in range(M): # Iterate over each transaction
    trans_id = input("Enter transaction ID: ").strip()

    print("Enter the number of payers and the number of splitters for transaction", i + 1, ":")
    n_pyr, n_spl = map(int, input().split())

    if not (1 <= n_pyr + n_spl <= 50): # Check if the sum of payers and splitters falls within the specified constraint
        print("Invalid input for the number of payers and splitters. Please ensure 1 <= len(paid_by) + len(split_as) <= 50.")
        exit()

    paid_by = []   # Store details of individuals who paid
    split_as = []  # Store details of amounts to split among individuals

    print("Enter details for each payer:") # Iterate over each payer in the transaction
    for _ in range(n_pyr):
        payer, amt_paid = map(int, input().split())
        paid_by.append([payer, amt_paid])

    print("Enter details for each splitter:") # Iterate over each splitter in the transaction
    for _ in range(n_spl):
        splitter, amt_split = map(int, input().split())
        split_as.append([splitter, amt_split])

    trans_list.append({'transaction_id': trans_id, 'paid_by': paid_by, 'split_as': split_as}) # Append transaction details to the list

# Main application
solve(N, trans_list)
```

```
 For all data inputs follow this format (1 60)
 Enter the number of people in the group (n) and the number of transactions recorded (m):
 6 5
 Enter details for each transaction:
 Enter transaction ID: #itsmylife
 Enter the number of payers and the number of splitters for transaction 1 :
```

```
2 3
Enter details for each payer:
1 25
3 15
Enter details for each splitter:
4 10
5 25
6 5
Enter transaction ID: #itsnow
Enter the number of payers and the number of splitters for transaction 2 :
1 4
Enter details for each payer:
4 100
Enter details for each splitter:
1 25
2 25
3 25
4 25
Enter transaction ID: #ornever
Enter the number of payers and the number of splitters for transaction 3 :
2 2
Enter details for each payer:
5 30
3 10
Enter details for each splitter:
1 25
4 15
Enter transaction ID: #iaintgonna
Enter the number of payers and the number of splitters for transaction 4 :
1 3
Enter details for each payer:
2 150
Enter details for each splitter:
1 50
2 50
3 50
Enter transaction ID: #liveforever
Enter the number of payers and the number of splitters for transaction 5 :
2 2
Enter details for each payer:
5 13
6 25
Enter details for each splitter:
4 25
1 13
5
1 2 75
1 4 13
3 4 12
3 5 18
3 6 20
```