```
def update_balances(trans_list):
    balances = {}

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

    return balances

def find_shortest_transfers(balances):
    payments = []

    while True: #loops until there are no more borrowers or lenders
        borrowers = sorted(filter(lambda x: balances[x] < 0, balances))
        lenders = sorted(filter(lambda x: balances[x] > 0, balances))

        if not borrowers or not lenders: # exit if no borrowers or lenders
            break

        borrower, lender = borrowers[0], lenders[0] #select first borrower &lender
        amt = min(abs(balances[borrower]), balances[lender]) #determine transfer amount

        payments.append([lender, borrower, amt])#add payment to list

        balances[borrower] += amt #update balance
        balances[lender] -= amt

    return payments

def print_payments(payments): #function to print the payments
    print(len(payments))
    for payment in payments:
        print(payment[1],payment[0],payment[2])

def main():
    N, M = map(int, input().split()) # get input for num of people (N) and num of transactions (M)

    if not (2 <= N <= 2 * 10**5) or not (1 <= M <= 5000): #input validation
        exit()

    trans_list = []

    for i in range(M): # for loop to get details for each transaction
        trans_id = input().strip() #map() converts input strings into integer
        n_payers, n_splits = map(int, input().split()) #which are assigned to variable payer and amt_paic

        if not (1 <= n_payers + n_splits <= 50): #check if the total number of payers and splitters is wi
            print()
            exit()

        paid_by = []
        split_as = []

        for _ in range(n_payers): #input details for payers
```

```
            payer, amt_paid = map(int, input().split()) #map converts input strings into integer
            paid_by.append([payer, amt_paid]) #which are assigned here, variables payer and amt_paid

        for _ in range(n_splits): #input details for splitters
            splitter, amt_split = map(int, input().split()) #map() converts inputs strings into integer
            split_as.append([splitter, amt_split]) #which are assigned to variables splitter and amt_spli

        trans_list.append({'transaction_id': trans_id, 'paid_by': paid_by, 'split_as': split_as})

    balances = update_balances(trans_list) # updates balance on transactions
    payments = find_shortest_transfers(balances) #update shortest path to settle balances to payments
    print_payments(payments) #print payments needed to pay

if __name__ == "__main__":
    main()
```

```
6 5
#itsmylife
2 3
1 25
3 15
4 10
5 25
6 5
#itsnow
1 4
4 100
1 25
2 25
3 25
4 25
#ornever
2 2
5 30
3 10
1 25
4 15
#iaintgonna
1 3
2 150
1 50
2 50
3 50
#liveforever
2 2
5 13
6 25
4 25
1 13
5
1 2 75
1 4 13
3 4 12
3 5 18
3 6 20
```