```python
import sys
import math
import numpy as np

class factory:
    def __init__(self, id):
        #nodes about
        self.id = id
        self.con = {}
        self.dist = {}

    def add_connection(self, id, node, dist):
        #connections between nodes
        self.con[id] = node
        self.dist[id] = dist

    def update_data(self, player, population, production):
        #reset node info
        self.player = player
        self.pop = population
        self.prod = production

#----------------------------------------------INIT-----------------------------------------
----

def route_move(node):
    #output variable
    move = ''
    #connections to out nodes
    our_nodes = [n for n in node.con.keys() if (node.con[n].player == 1)]
    if(our_nodes):
        #order by distance
        our_nodes.sort(key=lambda n: node.dist[n])
        #are there are enough to conqure
        if([n for n in our_nodes if nodes[n].pop > node.pop + 3]):
            move = 'MOVE %s %s %s;' % (our_nodes[0], node.id, node.pop + 2)
            #update population
            nodes[our_nodes[0]].pop -= node.pop + 2
        else:
            #sort by strength
            our_nodes.sort(key=lambda n: node.con[n].pop, reverse=True)
            #send as many as possible
            if(nodes[our_nodes[0]].pop > 2):
                move = 'MOVE %s %s %s;' % (our_nodes[0], node.id,
                nodes[our_nodes[0]].pop - 2)
                #update population
                nodes[our_nodes[0]].pop = 2
    return move


#----------------------------------------------INIT-----------------------------------------
----
#make array of factory nodes
node_count = int(input()) # the number of nodes
nodes = []
for id in range(node_count):
    nodes.append(factory(id))

#make a grid of bidirectional connections
for i in range(int(input())): # the number of links between nodes
    node_1, node_2, dist = [int(j) for j in input().split()]
    nodes[node_1].add_connection(node_2, nodes[node_2], dist)
    nodes[node_2].add_connection(node_1, nodes[node_1], dist)

# game loop
while True:
    entity_count = int(input())  # the number of entities (e.g. factories and troops)
```

```
    for i in range(entity_count):
        entity_id, entity_type, arg_1, arg_2, arg_3, arg_4, arg_5 = input().split()
        #update information for each turn
        if(entity_type == 'FACTORY'): #factory
            nodes[int(entity_id)].update_data(int(arg_1), int(arg_2), int(arg_3))

    moves = ''
    #get all neutral nodes sorted by distance
    neutral_nodes = sorted([n for n in nodes if n.player == 0], key = lambda n: n.prod,
    reverse=True)
    #are there any nodes in this group?
    if(neutral_nodes):
        for node in neutral_nodes:
            moves += route_move(node)

    #get all enemy nodes sorted by population
    enemy_nodes = sorted([n for n in nodes if n.player == -1], key = lambda n: n.pop)
    #are there any nodes in this group
    if(enemy_nodes):
        for node in enemy_nodes:
            moves += route_move(node)

    #if we can make a move, do so, otherwise wait
    if(moves != ''):
        print(moves[:-1])
    else:
        print("WAIT")

```