QMixer (PseudoCode):
Create Gym Env

Initialize Q-Network, Q_Target, MixNet, MixNet_target, Optimizer, replay_buffer
For episode in range(max_episodes):
    state= env.reset
    initialize q_network()
    while not all (done): action, hidden= q.sample(state) # sample using epsilon greedy
            next_state, reward, done, info = env.step(action) --> add to memory
            score += sum(reward)
            state= next_state
If memory > Threshold: #   Start Training:
      For in range(update_iter):
            s, a, r, s', done = Sample batch from memory [batch_size, chunk_size, n_agents]
            hidden= torch.zeros(batch_size), target = torch.zeros(batch_size)
            loss = 0
            For step_i in range(chunk_size):
                  q_out, hidden= q_net((state[step_i), hidden) # predicted q_values for all 5 actions for each agent
                  q_a = q_out(2, a[]) # actual actions taken at this time step_i
                  pred_q, next_mix_net_hidden   = mix_net(q_a, state[at time step_i], GRU_Cell(Mixnet_hidden)   # use all the Q-values for mixing

                  # repeated for the determining the q_target and hidden_target values
                  q', Target_hidden= q_net((s'[step_i), target.hidden) # predicted q_values for all 5 actions for each agent
                  max_q' = q'(2, max(dim=2)) # actual actions taken at this time step_i
                  q'_total, next_mix_net_hidden   = mix_net(max_q', s'[step_i], GRU_Cell(Mixnet_hidden)   # use all the Q-values for mixing
                   target_q = reward[step_i] + gamma* q'_total
                 Huber_loss (pred_q, target_q)
                 Optimizer.zerograd(), loss.backwards(), optimizer.step()