```javascript
// the game in normal form, a n dimensional matrix of payoff vectors.
// - it is indexed by the strategies to make the code clearer
game = {
  U: { L: [2, 6], C: [0, 4], R: [4, 4] },
  M: { L: [3, 3], C: [0, 0], R: [1, 5] },
  D: { L: [1, 1], C: [3, 5], R: [2, 3] },
}

// these are the parameters for each part of the question
// - player = the player for which to calculate best response
// - strategy = the probabilities in the other player's mixed strategy
questionParts = {
  a: { player: 1, strategy: [ 1/6, 1/3, 1/2 ] },
  b: { player: 2, strategy: [ 1/6, 1/3, 1/2 ] },
  c: { player: 1, strategy: [ 1/4, 1/8, 5/2 ] },
  d: { player: 1, strategy: [ 1/3, 1/3, 1/3 ] },
  e: { player: 2, strategy: [ 1/2, 1/2, 0/1 ] },
}

// This function calculates the best response
function bestResponse({ game, strategy, player }) {
  // this part of the code used to keep track of which strategies are best
  brSet = []
  maxPayoff = -99999
  function updateBR({ payoff, s }) {
    // add strategy to set of best responses if payoffs are equal:
    if (payoff === maxPayoff) brSet.push(s)

    // a new max payoff was found...
    // the set of best responses is the  current strategy:
    if (payoff > maxPayoff) {
      maxPayoff = payoff;
      brSet = [s]
    }
  }

  // This part of code calculates the payoffs
  if (player === 1) {
      // calculate player 1's payoff for each strategy s1 in S1
      S1 = ['U', 'M', 'D']
      for (s1 of S1) {
        const payoff =
          game[s1].L[0] * strategy[0] + // u1(s1,L) * probability s2 = L
          game[s1].C[0] * strategy[1] + // u1(s1,C) * probability s2 = C
          game[s1].R[0] * strategy[2]   // u1(s1,R) * probability s2 = R

        console.log(`u(${s1}, theta_2) = ${payoff}`) // log the payoff

        // keep track of which responses 's' are best
        updateBR({ payoff, s: s1 })
      }
    }

  // calculate player 2's payoff for reach strategy s2 in S2
  if (player === 2) {
    S2 = ['L', 'C', 'R']
    for (s2 of S2) {
      payoff =
        game.U[s2][1] * strategy[0] + // u2(U,s2) * probability s1 = U
        game.M[s2][1] * strategy[1] + // u2(M,s2) * probability s1 = M
        game.D[s2][1] * strategy[2]   // u2(D,s2) * probability s1 = D

      console.log(`u(theta_1, ${s2}) = ${payoff}`) // log the payoff
      updateBR({ payoff, s: s2 })

    }
  }
  return brSet
}

// execute the code for each part of the problem
for (letter of Object.keys(questionParts)) {
  console.log(`${letter})`)

  // get set of best responses and output
  br = bestResponse({ game, ...questionParts[letter] })
  console.log(`BR is {${br.join(',')}} \n`)
}

/*
 * output:

a)
u(U, theta_2) = 2.3333333333333335
u(M, theta_2) = 1
u(D, theta_2) = 2.166666666666667
BR is {U}

b)
u(theta_1, L) = 2.5
u(theta_1, C) = 3.1666666666666665
u(theta_1, R) = 3.833333333333333
BR is {R}

c)
u(U, theta_2) = 10.5
u(M, theta_2) = 3.25
u(D, theta_2) = 5.625
BR is {U}

d)
u(U, theta_2) = 2
u(M, theta_2) = 1.3333333333333333
u(D, theta_2) = 2
BR is {U,D}

e)
u(theta_1, L) = 4.5
u(theta_1, C) = 2
u(theta_1, R) = 4.5
BR is {L,R}

 */
```