```
// inputs in form of array
const input = ["F0", "1D", "9D", "E9", "E7"];

// hex to binary: https://gist.github.com/faisalman/4213592
const ConvertBase = function(num) {
  return {
    from: function(baseFrom) {
      return {
        to: function(baseTo) {
          return parseInt(num, baseFrom).toString(baseTo);
        }
      };
    }
  };
};

// hexadecimal to binary
ConvertBase.hex2bin = function(num) {
  return ConvertBase(num)
    .from(16)
    .to(2);
};

const inputBin = input.map(elem => {
  let result = ConvertBase.hex2bin(elem);
  // pad it to make 8 character string
  let padded = result.padStart(8, "0");
  // display padded string
  console.log(padded);
```

```
  const firstRow = padded
    .slice(0, 4)
    .split("")
    .map(x => parseInt(x, 0));

  const secondRow = padded
    .slice(4)
    .split("")
    .map(x => parseInt(x, 0));
  let paddedArray = [firstRow, secondRow];
  return paddedArray;
});
// display the matrix
console.log(inputBin);

const evalG4 = function(bAr) {
  let out = "";
  if (bAr[0][0] && bAr[0][1] && bAr[0][2] && bAr[0][3]) {
    out += "B";
    bAr[0][0] = 0;
    bAr[0][1] = 0;
    bAr[0][2] = 0;
    bAr[0][3] = 0;
  }
  if (bAr[1][0] && bAr[1][1] && bAr[1][2] && bAr[1][3]) {
    out += "~B";
    bAr[1][0] = 0;
    bAr[1][1] = 0;
    bAr[1][2] = 0;
```

```
  bAr[1][3] = 0;
}
if (bAr[0][2] && bAr[0][3] && bAr[1][2] && bAr[1][3]) {
  out += "~A";
  bAr[0][2] = 0;
  bAr[0][3] = 0;
  bAr[1][2] = 0;
  bAr[1][3] = 0;
}
if (bAr[0][0] && bAr[0][1] && bAr[1][0] && bAr[1][1]) {
  out += "A";
  bAr[0][0] = 0;
  bAr[0][1] = 0;
  bAr[1][0] = 0;
  bAr[1][1] = 0;
}
if (bAr[0][1] && bAr[0][2] && bAr[1][1] && bAr[1][2]) {
  out += "C";
  bAr[0][1] = 0;
  bAr[0][2] = 0;
  bAr[1][1] = 0;
  bAr[1][2] = 0;
}
if (bAr[0][0] && bAr[0][3] && bAr[1][0] && bAr[1][3]) {
  out += "~C";
  bAr[0][0] = 0;
  bAr[0][3] = 0;
  bAr[1][0] = 0;
  bAr[1][3] = 0;
```

```
  }
  return out;
};
const evalG2 = function(bAr) {
  let out = "";
  const resultValues = [["AB", "BC", "~AB"], ["A~B", "~BC", "~A~B"]];
  const resultValues2 = ["A~C", "AC", "~AC", "~A~C"];

  for (let i = 0; i < 2; i++) {
    for (let j = 0; j < 3; j++) {
      if (bAr[i][j] === 1 && bAr[i][j + 1] === 1) {
        out += `+${resultValues[i][j]}`;
        bAr[i][j] = 0;
        bAr[i][j + 1] = 0;
      }
    }
  }
  for (let i = 0; i < 1; i++) {
    for (let j = 0; j < 4; j++) {
      if (bAr[i][j] === 1 && bAr[i + 1][j] === 1) {
        out += `+${resultValues2[j]}`;
        bAr[i][j] = 0;
        bAr[i + 1][j] = 0;
      }
    }
  }
  if (bAr[0][0] === 1 && bAr[0][3] === 1) {
    out += "+" + "B~C";
    bAr[0][0] = 0;
```

```
    bAr[0][3] = 0;
  }
  if (bAr[1][0] === 1 && bAr[1][3] === 1) {
    out += "+" + "~B~C";
    bAr[1][0] = 0;
    bAr[1][3] = 0;
  }
  return out.replace(/^\++/, "");
};
const evalG1 = function(bAr) {
  let out = "";
  const resultValues = [
    ["AB~C", "ABC", "~ABC", "~AB~C"],
    ["A~B~C", "A~BC", "~A~BC", "~A~B~C"]
  ];
  for (let i = 0; i < 2; i++) {
    for (let j = 0; j < 4; j++) {
      if (bAr[i][j] === 1) {
        out += `+${resultValues[i][j]}`;
        bAr[i][j] = 0;
      }
    }
  }
  return out.replace(/^\++/g, "");
};

const finalOut = inputBin.map(row => {
  let result = evalG4(row);
  let resultG2 = evalG2(row);
```

```
  let resultG1 = evalG1(row);
  // filter blanks
  let out = [result, resultG2, resultG1].filter(elem => elem !== "");
  return out.join("+");
});
// Print final result
console.log("Results:");

input.forEach((elem, index) => {
  console.log(`${elem} → ${finalOut[index]}`);
});
```