#include *<bits/stdc++.h>*

#include *<ios>*

class Point2D {

public:

  float x, y;

  bool isCorrectSystem;

  Point2D(float *set\_x*, float *set\_y*) {

    x = set\_x;

    y = set\_y;

    isCorrectSystem = true;

  }

  Point2D return\_itself() { return Point2D(x, y); }

  Point2D rectangular\_to\_polar() {

    float r = sqrt(x \* x + y \* y);

    float phi;

    if (x > 0 && y >= 0) {

      phi = atan(y / x);

    } else if (x > 0 && y < 0) {

      phi = atan(y / x) + 2 \* M\_PI;

    } else if (x < 0) {

      phi = atan(y / x) + M\_PI;

    } else if (x == 0 && y > 0) {

      phi = M\_PI / 2;

    } else if (x == 0 && y < 0) {

      phi = -M\_PI / 2;

    } else {

      std::cout << *"x and y are equal to zero - WRONG"*;

      return return\_itself();

    }

    return Point2D(r, phi);

  }

  Point2D polar\_to\_rectangular() {

    float new\_x = x \* cos(y \* M\_PI / 180);

    float new\_y = x \* sin(y \* M\_PI / 180);

    return Point2D(new\_x, new\_y);

  }

  Point2D detecting\_system(std::string *syst\_to\_switch*,

                           std::string *current\_syst*) {

    if (syst\_to\_switch == current\_syst) {

      return return\_itself();

    }

    if (syst\_to\_switch == *"polar"*) {

      return rectangular\_to\_polar();

    }

    if (syst\_to\_switch == *"rectangular"*) {

      return polar\_to\_rectangular();

    } else {

      std::cout << *"Wrong system\n"*;

      isCorrectSystem = false;

      return return\_itself();

    }

  }

};

class Point3D {

public:

  float x, y, z;

  bool isCorrectSystem;

  Point3D(float *set\_x*, float *set\_y*, float *set\_z*) {

    x = set\_x;

    y = set\_y;

    z = set\_z;

    isCorrectSystem = true;

  }

  Point3D rectangular\_to\_spherical() { *// done*

    float radius\_vector = sqrt(x \* x + y \* y + z \* z); *// (x, y, z)*

    float teta = acos(z / radius\_vector);

    float phi = atan(y / x);

    return Point3D(radius\_vector, teta, phi);

  }

  Point3D rectangular\_to\_cylindrical() { *// done*

    float p = sqrtf(x \* x + y \* y);

    float phi = atan(y / x);

    return Point3D(p, phi, z);

  }

  Point3D spherical\_to\_rectangular() { *// done*

    float new\_x =

        x \* sin(z \* M\_PI / 180) \* cos(y \* M\_PI / 180); *// (p, phi, teta)*

    float new\_y = x \* sin(z \* M\_PI / 180) \* sin(y \* M\_PI / 180);

    float new\_z = x \* cos(z \* M\_PI / 180);

    return Point3D(new\_x, new\_y, new\_z);

  }

  Point3D spherical\_to\_cylindrical() { *// done*

    float r = x \* sin(z \* M\_PI / 180); *// (p, phi, teta)*

    float phi = y;

    float new\_z = x \* cos(z \* M\_PI / 180);

    return Point3D(r, phi, new\_z);

  }

  Point3D cylindrical\_to\_spherical() { *// done*

    float r = sqrt(x \* x + z \* z); *// (r, phi, z)*

    float teta = y;

    float phi = atan(x / z);

    return Point3D(r, teta, phi);

  }

  Point3D cylindrical\_to\_rectangular() { *// done*

    float new\_x = x \* cos(y \* M\_PI / 180); *// (r, phi, z)*

    float new\_y = x \* sin(y \* M\_PI / 180);

    float new\_z = y;

    return Point3D(new\_x, new\_y, new\_z);

  }

  Point3D return\_itself() { return Point3D(x, y, z); }

  Point3D to\_spherical(std::string *current\_system*) {

    if (current\_system == *"cylindrical"*) {

      return cylindrical\_to\_spherical();

    } else if (current\_system == *"rectangular"*) {

      return rectangular\_to\_spherical();

    } else {

      std::cout << *"Wrong current system\n"*;

      return return\_itself();

    }

  }

  Point3D to\_cylindrical(std::string *current\_system*) {

    if (current\_system == *"spherical"*) {

      return spherical\_to\_cylindrical();

    } else if (current\_system == *"rectangular"*) {

      return rectangular\_to\_cylindrical();

    } else {

      std::cout << *"Wrong current system\n"*;

      return return\_itself();

    }

  }

  Point3D to\_rectangular(std::string *current\_system*) {

    if (current\_system == *"spherical"*) {

      return spherical\_to\_rectangular();

    } else if (current\_system == *"cylindrical"*) {

      return cylindrical\_to\_rectangular();

    } else {

      std::cout << *"Wrong current system\n"*;

      isCorrectSystem = false;

      return return\_itself();

    }

  }

  Point3D detecting\_system(std::string *system\_to\_switch*,

                           std::string *current\_system*) {

    if (system\_to\_switch == current\_system) {

      return return\_itself();

    } else if (system\_to\_switch == *"spherical"*) {

      return to\_spherical(current\_system);

    } else if (system\_to\_switch == *"cylindrical"*) {

      return to\_cylindrical(current\_system);

    } else if (system\_to\_switch == *"rectangular"*) {

      return to\_rectangular(current\_system);

    } else {

      std::cout << *"Wrong switching system\n"*;

      isCorrectSystem = false;

      return return\_itself();

    }

  }

};

std::string toLower(std::string *str*) {

  transform(str.begin(), str.end(), str.begin(), ::tolower);

  return str;

}

float safeFloatInput(const std::string &*prompt*) {

  float value;

  while (true) {

    std::cout << prompt;

    if (!(std::cin >> value)) {

      std::cerr << *"Not a number, try again.\n"*;

      std::cin.clear(); *// Очистка флага ошибки*

      std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), *'\n'*);

    } else {

      return value;

    }

  }

}

void for\_three\_dim() {

  std::cout << *"Spherical (r, phi, teta)\nCylindrical (p, phi, z)\nRectangular "*

*"(x, y, z)\n\n"*;

  std::cout << *"Write coordinates:\n"*;

  float a = safeFloatInput(*"Enter first coordinate: "*);

  float b = safeFloatInput(*"Enter second coordinate: "*);

  float c = safeFloatInput(*"Enter third coordinate: "*);

  Point3D point(a, b, c);

  std::cout << *"What system do you use? (type quit to quit)\n"*;

  std::string current\_system;

  std::cin >> current\_system;

  current\_system = toLower(current\_system);

  if (current\_system == *"quit"*) {

    std::cout << *"Bye\n"*;

    return;

  }

  std::cout << *"What system do you want to switch?\n"*;

  std::string switch\_to\_syst;

  std::cin >> switch\_to\_syst;

  switch\_to\_syst = toLower(switch\_to\_syst);

  if (switch\_to\_syst == *"quit"*) {

    std::cout << *"Bye\n"*;

    return;

  }

  point = point.detecting\_system(switch\_to\_syst, current\_system);

  if (!point.isCorrectSystem) {

    return;

  }

  std::cout << *"What's the rounding factor?\n"*;

  double rounding;

  std::cin >> rounding;

  if (switch\_to\_syst == *"spherical"*) {

    std::cout << *"("* << *"r: "* << std::setprecision(rounding) << std::fixed

              << point.x << *", "* << *"teta: "* << std::setprecision(rounding)

              << std::fixed << point.y << *", "*

              << *"phi: "* << std::setprecision(rounding) << std::fixed

              << point.z \* 180 / M\_PI << *")"* << std::endl;

  }

  if (switch\_to\_syst == *"cylindrical"*) {

    std::cout << *"("* << *"r: "* << std::setprecision(rounding) << std::fixed

              << point.x << *", "* << *"phi: "* << std::setprecision(rounding)

              << std::fixed << point.y << *", "*

              << *"z: "* << std::setprecision(rounding) << std::fixed << point.z

              << *")"* << std::endl;

  }

  if (switch\_to\_syst == *"rectangular"*) {

    std::cout << *"("* << *"x: "* << std::setprecision(rounding) << std::fixed

              << point.x << *", "* << *"y: "* << std::setprecision(rounding)

              << std::fixed << point.y << *", "*

              << *"z: "* << std::setprecision(rounding) << std::fixed << point.z

              << *")"* << std::endl;

  }

}

void for\_two\_dim() {

  std::cout << *"1. Polar (r, phi)\n2. Rectangular (x, y)\n\n"*;

  std::cout << *"Write coordinates:\n"*;

  float a = safeFloatInput(*"Enter first coordinate: "*);

  float b = safeFloatInput(*"Enter second coordinate: "*);

  Point2D point(a, b);

  std::cout << *"What system do you use? (type quit to quit)\n"*;

  std::string current\_system;

  std::cin >> current\_system;

  current\_system = toLower(current\_system);

  if (current\_system == *"quit"*) {

    std::cout << *"Bye\n"*;

    return;

  }

  std::cout << *"What system do you want to switch?\n"*;

  std::string switch\_to\_syst;

  std::cin >> switch\_to\_syst;

  switch\_to\_syst = toLower(switch\_to\_syst);

  if (switch\_to\_syst == *"quit"*) {

    std::cout << *"Bye\n"*;

    return;

  }

  point = point.detecting\_system(switch\_to\_syst, current\_system);

  if (!point.isCorrectSystem) {

    return;

  }

  std::cout << *"What's the rounding factor?\n"*;

  double rounding;

  std::cin >> rounding;

  if (switch\_to\_syst == *"polar"*) {

    std::cout << *"("* << *"r: "* << std::setprecision(rounding) << std::fixed

              << point.x << *", "* << *"phi: "* << std::fixed

              << std::setprecision(rounding) << point.y \* 180 / M\_PI << *")"*

              << std::endl;

  }

  if (switch\_to\_syst == *"rectangular"*) {

    std::cout << *"("* << *"x: "* << std::setprecision(rounding) << std::fixed

              << point.x << *", "* << *"y: "* << std::setprecision(rounding)

              << std::fixed << point.y << *")"* << std::endl;

  }

}

int main() {

  std::cout << *"2 or 3 dimensions?\n"*;

  short dims;

  std::cin >> dims;

  if (dims == 2) {

    for\_two\_dim();

  }

  if (dims == 3) {

    for\_three\_dim();

  }

  return 0;

}