7.

import java.util.ArrayList;  
import java.util.Scanner;  
  
public class powCal {  
 // 제곱수: 1,4,9,16,25,36,49,64,81,100,121,144,169,225,256,289,324,361,400  
 public static boolean isSqrt(int target){ // 만약 제곱수가 존재하면 true  
 double sqrtNum = Math.sqrt(target);  
 if(sqrtNum!=(int)sqrtNum){  
 return false;  
 }  
 return true;  
 }  
  
 public static void main(String[] args) {  
 // 어떤 수가 제곱수인지 판단하려면 제곱근이 정수인지를 검사해야한다  
 Scanner sc = new Scanner(System.in);  
 ArrayList<String> answer = new ArrayList<>();  
 for (int i = 0; i < 7; i++) {  
 String inp = sc.next();  
 if (inp.contains("(")){ // 만약 ( 를 포함한다면 n(k)를 변환하는거  
 String lSqrt = ""; // 타겟을 처음 공백으로 초기화  
 String sqrtCnt = ""; // 제곱수를 공백으로 초기화  
 int num = 0; // 제곱될 수  
 int start=0; // 제곱수가 시작되는 시점  
 // 문자열 조작으로 타겟과 제곱수를 구한다  
 for (int j = 0; j < inp.length(); j++) {  
 if(inp.charAt(j)=='(') {  
 start=j+1; // j 부터 끝-1 까지는 제곱수이다  
 break; // (를 만날때 까지 계속  
 }  
 lSqrt += inp.charAt(j);  
 }  
 for (int j = start; j < inp.length()-1; j++) {  
 sqrtCnt+=inp.charAt(j); // 마지막 (는 항상 마지막이기 때문에 start부터 length-1까지가 제곱수  
 }  
  
 num = Integer.parseInt(lSqrt);  
 for (int j = 0; j < Integer.parseInt(sqrtCnt); j++) {  
 num = (int) Math.pow(num,2);  
 }  
 answer.add(String.format("%d",num)); // 정답에 제곱되어진 수를 넣는다  
 }  
 else{ // 만약 n(k) 로 변환해야한다면  
 int target = Integer.parseInt(inp);  
 int cnt = 0;  
 if (isSqrt(target) && target>1) { // 1이하는 제곱수가 될 수 없기에 아래로  
 while (true) { // 더이상 제곱수가 없을때 까지 반복  
 target = (int) Math.sqrt(target);  
 cnt++; // 루트한 횟수를 cnt 에 저장한다  
 if (isSqrt(target) == false) {  
 break;  
 } // 탈출 조건은 최소 제곱근일때이다  
 }  
 }  
 answer.add(String.format("%d(%d)",target,cnt)); // 정답에 올바른 형태로 저장한다  
 }  
 for (String s : answer) {  
 System.out.println(s); // 리스트에 저장된 변환된 수를 반환한다  
 }  
 }  
 }  
}

리뷰: breakpoint 가 따로 요구사항에 나와있지 않아 예시대로 7번만 입력 받아 출력하는 형태로 코딩을 하였습니다. 문자열 조작을 통하여 제곱되어질 수와 제곱횟수를 찾아내는 것을 간소화 할 수 있을 것같은데 Java에 아직 능숙하지 않아 원시적인 방법으로 하나하나 조회하여 따로 구했습니다. 제곱수로 변환 같은 경우 isSqrt를 통하여 제곱이 되는지를 판별하였습니다. 이 또한 원시적으로 소수점 아래가 존재하는지를 통하여 true 또는 false 로 나누어 만약 true 라면 최소 제곱수가 나오도록 반복하여 루트된 횟수를 cnt에 저장하였습니다. 마지막으로 format 함수를 통하여 answer에 답을 넣었는데 변수를 너무 많이 쓴 점과 로직 자체가 올바른지는 아직도 명확하지 않습니다. 1 의 제곱수가 있다고 할시 무한 반복되기에 1보다 큰경우일때만 제곱수를 구하도록 하였습니다.

8.

import java.util.Scanner;  
  
public class clockDegree {  
 public static void main(String[] args) {  
 // 시계를 구현하고 분침, 시침간의 각도를 구하여 입력되는 A 와 각도가 같을시 리스트에 저장하도록 한다  
 // 초당 분침, 시침이 이동하는 것을 통해 초당 시침과 분침의 각도는 0.0083333 과 0.1 도 이다  
 // 하루는 총 86400초이기 때문에 이동안 분침, 시침이 증가되면서 이루는 각도의 차가 입력받은 값일때 answer에 담도록한다  
 Scanner sc = new Scanner(System.in);  
 int angle = sc.nextInt();  
 double hourD = 0;  
 double minD = 0;  
  
 for (int hour = 0; hour < 24; ++hour) {  
 for (int min = 0; min < 60; ++min) {  
 for (int sec = 1; sec < 60; ++sec) {  
 hourD += 1/120\*sec; // 매초 시침과 분침의 증가치  
 minD += 0.1\*sec;  
 if(hourD>=360) hourD = 0; // 만약 360도 이상이라면 0으로 초기화  
 if(minD>=360) minD = 0; // 분침또한  
 if((double) angle==Math.abs(hourD-minD)||(double) angle==360-Math.abs(hourD-minD)){  
 // 만약 각도가 시침각과 분침각의 차라면 , 단 차는 180을 넘어가면 360-차 이기 때문에 두가지 경우를 생각한다  
 System.out.println(String.format("%02d:%02d:%02d",hour,min,sec)); // 출력형식에 맞게 두자리로 출력하되 앞자리가 없다면 0으로 채워준다  
 }  
 }  
 }  
 }  
 }  
}

리뷰: 본래 구상했던 것은 반복문 한번으로 시, 분, 초 를 증가시켜 구하는 것이였는데 60분마다, 60초마다 if 로 조건식을 덕지덕지 붙이니 오히려 가독성이 떨어지는 것 같아 직관적으로 3중 for문으로 시침각과 분침각이 이루는 각도를 구하였습니다. 간단한 수학적 지식으로 1초당 움직이는 시침의 각도와 분침의 각도를 이용하여 360도를 초과할시 0으로 초기화 되도록 하였습니다. 분침각과 시침각이 이루는 각도중 기준이 되는 각을 알기 어려웠기에 절대값을 적용하여 이 각도의차가 목표 각도와 동일할시 시분초가 출력되도록 하였습니다. 그리고 문제의 요구상 0초 일때는 증가하는 것이 없어야 하므로 1초부터 시작하도록 하였습니다. 조건 하나라도 잘못 설정할시 답이 출력되지 않아 제약조건에 대한 중요성을 다시 한번 알게 되어 좋았던것같습니다.