2. Write a complete syntactically correct java program that prompts the user for two double values \(x\) and \(y\), then prints out the value of the expression \(\sqrt{x} + \sqrt{y}\). Include all necessary import statements, a class definition and \texttt{main()} function. You may give the class any valid name. Specify the name of the file that contains your program in a one-line comment at the beginning of the program. (Hint: search the documentation for the class \texttt{java.lang.Math} for a function called \texttt{sqrt()} and use it.)

\textbf{Solution:}

// Problem2.java

import java.util.Scanner;

class Problem2{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        double x, y, z;

        System.out.print("Enter two doubles: ");
        x = sc.nextDouble();
        y = sc.nextDouble();

        z = Math.sqrt(x) + Math.sqrt(y);
        System.out.println(z);
    }
}
7. Determine the output of the following Java program for any and all possible inputs by the user. In other words, describe the program’s operation with sufficient generality so that, given any sequence of key presses by the user, one could predict the printed result.

```java
// Problem7.java
import java.util.Scanner;

class Problem7{
    public static void main( String[] args ){
        Scanner sc = new Scanner(System.in);
        int a = 0;
        double x=0, y=0, z=0;
        while( a<30 ){
            while(true){
                while(!sc.hasNextDouble()){
                    sc.next();
                }
                if( a<10 ){
                    x = sc.nextDouble();
                    if( x>0 ) break;
                }else if( a<20 ){
                    y = sc.nextDouble();
                    if( y<0 ) break;
                }else{
                    z = sc.nextDouble();
                    if( z>0 ) break;
                }
                a += 10;
            }
        }
        System.out.println("x = " + x + ", y = " + y + ", z = " + z);
    }
}
```

Solution:
The program will scan tokens in the input stream until it finds a positive double, then store it in \( x \). It will then scan up to the next negative double and store that in \( y \). It will then scan up to the next positive double and place that in \( z \). Finally \( x \), \( y \) and \( z \) are printed out in order.
8. Write a complete Java program that reads 5 integers entered by the user, and prints them out in increasing order. Your program should continually prompt for integer input, discarding tokens that are not integers, until 5 integers have been entered by the user. Hint: study the `CheckInput` sequence of examples and `CollatzProblem.java` to see how to filter user input, and emulate the examples `Sort3.java` and `Sort4.java` to sort the 5 integers in increasing order.

Solution:
```java
import java.util.Scanner;
class Problem8{
    public static void main( String[] args ){
        int a=0, b=0, c=0, d=0, e=0, temp, x;
        Scanner sc = new Scanner(System.in);
        // get a, b, c, d, e from user
        for(int i=0; i<5; i++){  // 5 times
            System.out.print("Enter an integer: ");
            while(true){
                if( !sc.hasNextInt() ){  // if the next token is not int
                    sc.next();            // throw it away
                }else{                   // otherwise
                    x = sc.nextInt();     // extract the int
                    break;                // and keep it
                }
            // if its not an int, ask nicely and try again
            System.out.print("Please enter an integer: ");
            }
            // we now have an int in x, so assign it to the right variable
            if(i==0)      a = x;
            else if(i==1) b = x;
            else if(i==2) c = x;
            else if(i==3) d = x;
            else if(i==4) e = x;
            // We have to ask which iteration we are on to figure out what variable
            // to assign to. Later we'll learn a more elegant way to do this
            // with arrays.
        }
        // sort a, b, c, d, e in increasing order
        if( a>b ){ temp=a; a=b; b=temp; }
        if( b>c ){ temp=b; b=c; c=temp; }
        if( c>d ){ temp=c; c=d; d=temp; }
        if( d>e ){ temp=d; d=e; e=temp; }
        if( a>b ){ temp=a; a=b; b=temp; }
        if( b>c ){ temp=b; b=c; c=temp; }
        if( c>d ){ temp=c; c=d; d=temp; }
        if( a>b ){ temp=a; a=b; b=temp; }
        if( b>c ){ temp=b; b=c; c=temp; }
        if( a>b ){ temp=a; a=b; b=temp; }
        // print numbers in increasing order
        System.out.println(a + " " + b + " " + c + " " + d + " " + e);
    }
}
```
Here's a solution to problem #9 that does not use functions. Instead we just write code in function main() that creates the desired table. Exercise: read documentation on the printf() function in Java.io.PrintStream.

```
// Problem9NoFunctions.java
import java.util.Scanner;

class Problem9NoFunctions{
    public static void main( String[] args ){
        int n;
        System.out.println(
            "Integer     Square      Cube        Quartic     Quintic"
        );
        System.out.println(
            "-------------------------------------------------");
        for(n=1; n<=25; n++){
            System.out.printf("%-12d", n);
            System.out.printf("%-12d", (int)Math.pow(n,2) );
            System.out.printf("%-12d", (int)Math.pow(n,3) );
            System.out.printf("%-12d", (int)Math.pow(n,4) );
            System.out.printf("%-12d
", (int)Math.pow(n,5) );
        }
    }
}
```