**Terminology**

**Debugging.** Identifying and fixing errors in computer code.

A **compile-time error** is an error that occurs when compiling the program.

```java
public class BadHelloWorld {
    public static void main(String[] args) {
        print "Hello, World"
    }
}
```

A **run-time error** is an error that occurs while the program is running.

```java
public class DivideByZero {
    public static void main(String[] args) {
        int x = 10;
        int y = 0;
        int z = x / y;
    }
}
```
DEBUGGING

- bug hunt
- live coding
What’s the bug?
Bug 1 (integer division)

What is the value of the variable \( x \) at the end of the computation?

```java
int numerator = 3;
int denominator = 5;
double x = (double) (numerator/denominator);
```

A. 0.0
B. 0.59999999999999997779553950749686919152736663818359375
C. 0.6
D. compile-time error
E. run-time error

intended

```
(double) numerator / denominator;
```

or

```
1.0 * numerator / denominator;
```
Bug 2 (invoking method with incorrect argument types)

What is the values of $\text{max}$ at the end of the computation?

A. -4
B. 2
C. 4
D. compile-time error
E. run-time error

int $a = 2$;
int $b = 4$;
int $c = -4$;
int $\text{max} = \text{Math.max}(a, b, c)$;

error: no suitable method found for
$\text{max}(\text{int}, \text{int}, \text{int})$
method $\text{Math.max}(\text{int},\text{int})$
is not applicable
method $\text{Math.max}(\text{long},\text{long})$
is not applicable
method $\text{Math.max}(\text{float},\text{float})$
is not applicable
method $\text{Math.max}(\text{double},\text{double})$
is not applicable

intended
int $\text{max} = \text{Math.max}(\text{Math.max}(a, b), c)$;
Bug 3  (can’t chain comparison operators)

What does the following code fragment print?

```java
double humidity = 40.0;
if (40 <= humidity <= 60.0)
    System.out.println("confortable");
error: bad operand types for binary operator ‘<='
first type: boolean
second type: int
```

A. confortable
B. 40.0
C. nothing
D. compile-time error  
```java
intended
if (humidity >= 40.0 && humidity <= 60.0)
```
E. run-time error
What does the following code fragment print?

```java
int x = 0;
if (x = 0) System.out.println("zero");
else System.out.println("not zero");
```

error: Incompatible types
int cannot be converted to boolean

A. zero
B. not zero
C. nothing
D. compile-time error
E. run-time error
Bug 4b  (= instead of ==)

What does the following code fragment print?

```java
int x = -123;
boolean isPositive = (x > 0);
if (isPositive == true) System.out.println("positive");
else System.out.println("not positive");
```

A. "positive"
B. "not positive"
C. nothing
D. compile-time error
E. run-time error

intended
```
if (isPositive == true)
```
better style
```
if (isPositive)
```
Bug 5  *(digits 0 and 1 vs. letters 0 and I)*

What is the values of `result` at the end of the computation?

```java
int l = 0;
int 0 = 1;
int 100 = 2;
int result = 100 + 100 + 0*0 + 1*1 + 1001;
```

A. 1024  
B. 1104  
C. 1201  
D. 1202  
E. compile-time error  
F. *never write code like this! (the answer is 1104, not 1104)*
Bug 6  (using a keyword as a variable name)

What does the following code fragment print?

```java
int default = 18;
int x = 24;
boolean resetToDefault = true;
if (resetToDefault)
    x = default;
System.out.println(x);
```

A. zero
B. not zero
C. nothing
D. compile-time error
E. run-time error
Bug 6  *(using a keyword as a variable name)*

**Q.** How would I know that `default` is a reserved keyword?

**A.** Syntax highlighting provides a clue.

```java
int default = 18;
int x = 24;
boolean resetToDefault = true;
if (resetToDefault)
    x = default;
```
Why is it called "debugging"?

Admiral Grace Hopper’s operational logbook for the Harvard Mark II computer.
Yale to change Calhoun College’s name to honor Grace Murray Hopper

February 11, 2017
What does the following code fragment print?

```java
int x = -123;
if (x > 0) {
    System.out.println("positive");
}
```

A. "positive"
B. "not positive"
C. nothing
D. compile-time error
E. run-time error

Equivalent to:

```java
if (x > 0) {
    System.out.println("positive");
}
```
Bug 7b (spurious semicolon)

What does the following code fragment print?

```
int x = 99;
while (x > 0); {
    x = x / 2;
}
System.out.println(x);
```

A. 0  
B. 1  
C. 99  
D. nothing  
E. compile-time error
What is the value of the variable \textit{force} at the end of the computation?

```
double G = 1.0; // in some other universe
double mass1 = 1.0;
double mass2 = 2.0;
double r = 5.0;
double \textit{force} = G \times \text{mass}1 \times \text{mass}2 / r \times r;
```

<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
<th>Derived Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.</td>
<td>0.0</td>
<td>intended</td>
</tr>
<tr>
<td>B.</td>
<td>0.04</td>
<td>( G \times \text{mass}1 \times \text{mass}2 / (r \times r) )</td>
</tr>
<tr>
<td>C.</td>
<td>0.08</td>
<td>equivalent to</td>
</tr>
<tr>
<td>D.</td>
<td>2.0</td>
<td>((G \times \text{mass}1 \times \text{mass}2 / r) \times r )</td>
</tr>
<tr>
<td>E.</td>
<td>NaN</td>
<td></td>
</tr>
</tbody>
</table>

Multiplication and division have the same precedence; the associativity is left-to-right.
Bug 9  *(ignoring return value of method)*

Which is the value of the variable \( x \) at the end of the computation?

```java
double x = 180.0;
Math.toRadians(x);
```

A. 0.0  
B. 1.5707963267948966  
C. 3.141592653589793  
D. 180.0  
E. 10313.2

*intended*

```java
double x = 180.0;
x = Math.toRadians(x);
```
“If debugging is the process of removing bugs, then programming must be the process of putting them in.” — Edsger Dijkstra

“Everyone knows that debugging is twice as hard as writing a program in the first place. So if you’re as clever as you can be when you write it, how will you ever debug it?” — Brian Kernighan
DEBUGGING

• bug hunt
• live coding
Debugging a complete program

**Goal.** Write a program that reads an integer command-line argument $n$ and prints $n! = 1 \times 2 \times 3 \times \ldots \times n$.

<table>
<thead>
<tr>
<th>$n$</th>
<th>$n!$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>24</td>
</tr>
<tr>
<td>5</td>
<td>120</td>
</tr>
<tr>
<td>6</td>
<td>720</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
</tr>
</tbody>
</table>

**factorial function**
Bug 0 (code not in body of main)

**Goal.** Write a program that reads an integer command-line argument $n$ and prints $n! = 1 \times 2 \times 3 \times \ldots \times n$.

```java
public class Factorial {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        for (int i = 0; i < n; i++)
            product = product * i;
        }
        System.out.println(product);
    }

```
Bug 1 (undeclared variable)

Goal. Write a program that reads an integer command-line argument $n$ and prints $n! = 1 \times 2 \times 3 \times \ldots \times n$. 

```java
public class Factorial {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        for (int i = 0; i < n; i++)
            product = product * i;
        System.out.println(product);
    }
}
```

```
error: cannot find symbol
  product = product * i;
     ^
symbol: variable product

2 errors
```
Goal. Write a program that reads an integer command-line argument $n$ and prints $n! = 1 \times 2 \times 3 \times \cdots \times n$.

```java
public class Factorial {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        int product;
        for (int i = 0; i < n; i++)
            product = product * i;
        System.out.println(product);
    }
}
```

error: variable product might not have been initialized
```java
product = product * i;
```
Goal. Write a program that reads an integer command-line argument \( n \) and prints \( n! = 1 \times 2 \times 3 \times \ldots \times n \).

```java
public class Factorial {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        int product;
        for (int i = 0; i < n; i++) {
            product = 0; // error
            product = product * i;
            System.out.println(product);
        }
    }
}
```

```
error: cannot find symbol
    product = product * i;
          ^
symbol:   variable i
1 error
```
Bug 4 (initialization in wrong place)

**Goal.** Write a program that reads an integer command-line argument \( n \) and prints \( n! = 1 \times 2 \times 3 \times \ldots \times n \).

```java
public class Factorial {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        int product;
        for (int i = 0; i < n; i++) {
            product = 0;
            product = product * i;
        }
        System.out.println(product);
    } // Error: variable product might not have been initialized
}
```

```
error: variable product might not have been initialized
   System.out.println(product);
       ^
1 error
```
Bug 5 (forgot to type command-line arguments)

Goal. Write a program that reads an integer command-line argument $n$ and prints $n! = 1 \times 2 \times 3 \times \ldots \times n$.

```java
public class Factorial {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        int product = 1;
        for (int i = 0; i < n; i++) {
            product = product * i;
        }
        System.out.println(product);
    }
}
```

% java Factorial
```
Exception in thread "main"
java.lang.ArrayIndexOutOfBoundsException: 0
  at Factorial.main(Factorial.java:3)
```
Bug 6 (off-by-one error)

**Goal.** Write a program that reads an integer command-line argument $n$ and prints $n! = 1 \times 2 \times 3 \times \ldots \times n$.

```java
public class Factorial {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        int product = 1;
        for (int i = 0; i < n; i++) {
            product = product * i;
        }
        System.out.println(product);
    }
}
```

```
% java Factorial 3
0

% java Factorial 5
0
```
Hooray!

**Goal.** Write a program that reads an integer command-line argument \( n \) and prints \( n! = 1 \times 2 \times 3 \times \ldots \times n \).

```java
public class Factorial {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        int product = 1;
        for (int i = 1; i <= n; i++) {
            product = product * i;
        }
        System.out.println(product);
    }
}
```

% java Factorial 3
6

% java Factorial 5
120