XJTLU | Computing

CPT108: Data Structures and Algorithms

Semester 2, 2023-24

Supplementary note: Code tracing

> The exercises included in this supplementary note are *not* compulsory, and will *not* be graded. However, you are encouraged to finish this to test your programming and analytical skills.

Code tracing is a simulation of code execution in which you step through instructions and track the values of the variables.

When you hand-trace code or pseudocode, you write the names of the variables on a sheet of paper, mentally execute each step of the code and update the variables.

It is best to have the code written or printed on a sheet of paper. Use a marker, such as a paper clip or a dot with your pen, to mark the current line. Whenever a variable changes, cross out the old value and write the new value next to the old one. When a program produces output, also write down the output in another column.

Example 1: Code tracing with loop

Consider the code snippet below. What value is displayed?

```
int n = 1234;
int sum = 0;
while (n > 0) {
  int digit = n % 10;
  sum = sum + digit;
  n = n / 10;
}
System.out.println(sum);
```

In this snippet there are three variables: `n`, `sum` and `digit`, and an output. So, on the paper, we should have a table similar to the one below.

n	sum	digit	Output

Now we start tracing the code.

The first two variables are initialized with 1234 and 0 before entering the loop.

```
  1 int n = 1234;
→ 2 int sum = 0;
  3 while (n > 0) {
  4   int digit = n % 10;
  5   sum = sum + digit;
  6   n = n / 10;
  7 }
  8 System.out.println(sum);
```

n	sum	digit	Output
1234	0		

Because $n > 0$, so we enter the loop.

The variable `digit` is set to 4.

```
  1 int n = 1234;
  2 int sum = 0;
  3 while (n > 0) {
→ 4   int digit = n % 10;
  5   sum = sum + digit;
  6   n = n / 10;
  7 }
  8 System.out.println(sum);
```

n	sum	digit	Output
1234	0		
		4	

And subsequently the variable `sum` is set to `0+4` = `4`.

```
  1 int n = 1234;
  2 int sum = 0;
  3 while (n > 0) {
  4   int digit = n % 10;
→ 5   sum = sum + digit;
  6   n = n / 10;
  7 }
  8 System.out.println(sum);
```

n	sum	digit	Output
1234	~~0~~		
	4	4	

Then, in line 6, the value of `n` is updated to `1234/10` = `123`.

```
  1 int n = 1234;
  2 int sum = 0;
  3 while (n > 0) {
  4   int digit = n % 10;
  5   sum = sum + digit;
→ 6   n = n / 10;
  7 }
  8 System.out.println(sum);
```

n	sum	digit	Output
~~1234~~	~~0~~		
123	4	4	

Now, check the loop condition in line 3 again.

```
  1 int n = 1234;
  2 int sum = 0;
→ 3 while (n > 0) {
  4   int digit = n % 10;
  5   sum = sum + digit;
  6   n = n / 10;
  7 }
  8 System.out.println(sum);
```

n	sum	digit	Output
~~1234~~	~~0~~		
123	4	4	

Because `n` is still greater than zero, the code in the loop repeat again.

Now the variable `digit` becomes 3.

```
  int n = 1234;
  int sum = 0;
  while (n > 0) {
→   int digit = n % 10;
    sum = sum + digit;
    n = n / 10;
  }
  System.out.println(sum);
```

n	sum	digit	Output
~~1234~~	~~0~~		
123	4	~~4~~	
		3	

`sum` is set to 7

```
  int n = 1234;
  int sum = 0;
  while (n > 0) {
    int digit = n % 10;
→   sum = sum + digit;
    n = n / 10;
  }
  System.out.println(sum);
```

n	sum	digit	Output
~~1234~~	~~0~~		
123	~~4~~	~~4~~	
	7	3	

And `n` is set to 12.

```
  int n = 1234;
  int sum = 0;
  while (n > 0) {
    int digit = n % 10;
    sum = sum + digit;
→   n = n / 10;
  }
  System.out.println(sum);
```

n	sum	digit	Output
~~1234~~	~~0~~		
~~123~~	~~4~~	~~4~~	
12	7	3	

As `n` is greater than zero, the loop repeat again, and the values of the variables `digit`, `sum` and `n` are the set to 2, 9 and 1, respectively. (You need to trace it yourself this time!)

```
  int n = 1234;
  int sum = 0;
→ while (n > 0) {
    int digit = n % 10;
    sum = sum + digit;
    n = n / 10;
  }
  System.out.println(sum);
```

n	sum	digit	Output
~~1234~~	~~0~~		
~~123~~	~~4~~	~~4~~	
~~12~~	~~7~~	~~3~~	
1	9	2	

And in the next iteration, the values of the three variables above become 1, 10 and 0, respectively.

```
  int n = 1234;
  int sum = 0;
→ while (n > 0) {
    int digit = n % 10;
    sum = sum + digit;
    n = n / 10;
  }
  System.out.println(sum);
```

n	sum	digit	Output
~~1234~~	~~0~~		
~~123~~	~~4~~	~~4~~	
~~12~~	~~7~~	~~3~~	
~~1~~	~~9~~	~~2~~	
0	10	1	

Because n equals to zero, the condition in line 3 is now false. We continue with the statement after the loop (line 8).

```
   1  int n = 1234;
   2  int sum = 0;
   3  while (n > 0) {
   4    int digit = n % 10;
   5    sum = sum + digit;
   6    n = n / 10;
   7  }
→  8  System.out.println(sum);
```

n	sum	digit	Output
~~1234~~	~~0~~		
~~123~~	~~4~~	~~4~~	
~~12~~	~~7~~	~~3~~	
~~1~~	~~9~~	~~2~~	
0	10	1	10

The statement in line 8 is an output statement. So, the value of the variable *sum*, i.e., 10, will be sent to the output.

In essence, the snippet above is used to calculate the sum of digits in n. Operations of this kind are useful for verifying credit card numbers and other forms of ID numbers.

Consider what happens in each iteration:

- We set the last digit of n to variable digit
- We add the value of digit to sum
- We strip the last digit off n

The contents above show you how code tracing can give you an *insight* that you would not get if you simply ran the code.

Example 2: Code tracing with loop – Another example

Consider now the pseudocode below. What value is displayed?

```
int i = 0;
int sum = 0;
while (sum >= 10) {
  i++;
  sum = sum + i;
  System.out.println(i + ":" + sum);
}
```

In this example, there are two variables in the code snippet: i and sum, and an output.

The first two variables are initialized with 1234 and 0 before entering the loop.

```
  int i = 0;
→ int sum = 0;
  while (sum >= 10) {
    i++;
    sum = sum + i;
    System.out.println(i + ":" + sum);
  }
```

i	sum	Output
0	0	

However, as sum is *not* greater than or equal to 10, we are not entering into the loop and the program ends!

```
   1  int i = 0;
   2  int sum = 0;
→  3  while (sum >= 10) {
   4    i++;
   5    sum = sum + i;
   6    System.out.println(i + ":" + sum);
   7  }
```

i	sum	Output
0	0	

Therefore, there is <u>*no*</u> output in this case.

Suppose now we modify the code snippet as follow (the condition of the while-loop in line 3 is changed to sum<10).

```
1  int i = 0;
2  int sum = 0;
3  while (sum < 10) {
4    i++;
5    sum = sum + i;
6    System.out.println(i + ": " + sum);
7  }
```

In this case, because sum $<$ 10, we enter the loop.

The variable i is then incremented by 1.

```
   1  int i = 0;
   2  int sum = 0;
   3  while (sum < 10) {
→  4    i++;
   5    sum = sum + i;
   6    System.out.println(i + ": " + sum);
   7  }
```

i	sum	Output
~~0~~	0	
1		

Next, the variable sum is set to sum+1=1.

```
   1  int i = 0;
   2  int sum = 0;
   3  while (sum < 10) {
   4    i++;
→  5    sum = sum + i;
   6    System.out.println(i + ": " + sum);
   7  }
```

i	sum	Output
~~0~~	~~0~~	
1	1	

Then, we send 1:1 (i+":"+sum) to the output.

```
   1  int i = 0;
   2  int sum = 0;
   3  while (sum < 10) {
   4    i++;
   5    sum = sum + i;
→  6    System.out.println(i + ": " + sum);
   7  }
```

i	sum	Output
~~0~~	~~0~~	
1	1	1:1

Because sum is less than 10, so the code in the loop repeat again.

Now, i becomes 2 and sum becomes 3.

```
  int i = 0;
  int sum = 0;
  while (sum < 10) {
    i++;
→   sum = sum + i;
    System.out.println(i + ": " + sum);
  }
```

i	sum	Output
~~0~~	~~0~~	
~~1~~	~~1~~	1:1
2	3	

And we send 2:3 to the output.

```
  int i = 0;
  int sum = 0;
  while (sum < 10) {
    i++;
    sum = sum + i;
→   System.out.println(i + ": " + sum);
  }
```

i	sum	Output
~~0~~	~~0~~	
~~1~~	~~1~~	1:1
2	3	1:1 2:3

As sum still less than 10, the loop repeat again, and the values of the variables i and sum become 3 and 6, respectively, and 3:6 will be sent to the output.

```
  int i = 0;
  int sum = 0;
→ while (sum < 10) {
    i++;
    sum = sum + i;
    System.out.println(i + ": " + sum);
  }
```

i	sum	Output
~~0~~	~~0~~	
~~1~~	~~1~~	1:1
~~2~~	~~3~~	1:1 2:3
3	6	1:1 2:3 3:6

And in the next iteration, the two variables above become 4 and 10, and 4:10 will be sent to the output.

```
  int i = 0;
  int sum = 0;
→ while (sum < 10) {
    i++;
    sum = sum + i;
    System.out.println(i + ": " + sum);
  }
```

i	sum	Output
~~0~~	~~0~~	
~~1~~	~~1~~	1:1
~~2~~	~~3~~	1:1 2:3
~~3~~	~~6~~	1:1 2:3 3:6
4	10	1:1 2:3 3:6 4:10

Since sum is now 10 and the condition is no longer valid, the loop end and the program terminates.

Hence, after running the snippet, we have i $= 3$, sum $= 10$, and output $\boxed{\begin{array}{l}1{:}1\\2{:}3\\3{:}6\\4{:}10\end{array}}$.

Example 3: Code tracing with function call

Consider the code snippet below. What value is displayed?

```java
public static double f(double a, double b) {
  double sum;
  sum = a + b;
  return sum;
}

public static void main(String[] arguments) {
  double dOne = 1.5;
  double dTwo = 3.1;
  double total = 0;
  for (int i = 0; i <= 3; i++) {
    if (i < dOne) {
      total = total + f(dOne, dTwo);
    } else {
      total = total + f(total, dOne);
    }
  }
  System.out.println("total=" + total);
}
```

In this snippet there are three variables: `dOne`, `dTwo` and `total`, and an Output. So, on the paper, we should have a table similar to the one below.

main()			
dOne	dTwo	total	Output

The three variables: `dOne`, `dTwo` and `total` are initialized with 1.5, 3.1 and 0, respectively, before entering the `for`-loop.

```java
  public static double f(double a, double b) {
    double sum;
    sum = a + b;
    return sum;
  }

  public static void main(String[] arguments) {
    double dOne = 1.5;
    double dTwo = 3.1;
→   double total = 0;
    for (int i = 0; i <= 3; i++) {
      if (i < dOne) {
        total = total + f(dOne, dTwo);
      } else {
        total = total + f(total, dOne);
      }
    }
    System.out.println("total=" + total);
  }
```

main()			
dOne	dTwo	total	Output
1.5	3.1	0	

Now we enter the `for`-loop. A new variable `i` (the *loop counter*) appears and is set to 0.

```
    1  public static double f(double a, double b) {
    2    double sum;
    3    sum = a + b;
    4    return sum;
    5  }
    6
    7  public static void main(String[] arguments) {
    8    double dOne = 1.5;
    9    double dTwo = 3.1;
   10    double total = 0;
→  11    for (int i = 0; i <= 3; i++) {
   12      if (i < dOne) {
   13        total = total + f(dOne, dTwo);
   14      } else {
   15        total = total + f(total, dOne);
   16      }
   17    }
   18    System.out.println("total=" + total);
   19  }
```

main()				
dOne	dTwo	total	i	Output
1.5	3.1	0		
			0	

Next, we need to check the condition in line 12.

```
    1  public static double f(double a, double b) {
    2    double sum;
    3    sum = a + b;
    4    return sum;
    5  }
    6
    7  public static void main(String[] arguments) {
    8    double dOne = 1.5;
    9    double dTwo = 3.1;
   10    double total = 0;
   11    for (int i = 0; i <= 3; i++) {
→  12      if (i < dOne) {
   13        total = total + f(dOne, dTwo);
   14      } else {
   15        total = total + f(total, dOne);
   16      }
   17    }
   18    System.out.println("total=" + total);
   19  }
```

main()				
dOne	dTwo	total	i	Output
1.5	3.1	0		
			0	

As `dOne` is 1.5 and is greater than `i`, so the condition in line 12 is satisfied. Therefore, we continue the execution with line 13.

```
    1  public static double f(double a, double b) {
    2    double sum;
    3    sum = a + b;
    4    return sum;
    5  }
    6
    7  public static void main(String[] arguments) {
    8    double dOne = 1.5;
    9    double dTwo = 3.1;
   10    double total = 0;
   11    for (int i = 0; i <= 3; i++) {
   12      if (i < dOne) {
→  13        total = total + f(dOne, dTwo);
   14      } else {
   15        total = total + f(total, dOne);
   16      }
   17    }
   18    System.out.println("total=" + total);
   19  }
```

main()				
dOne	dTwo	total	i	Output
1.5	3.1	0		
			0	

Now, we have a function call to the function `f` with two arguments `dOne` and `dTwo`.

So, in the function `f`, we have `a` $= 1.5$ and `b` $= 3.1$, and `sum` is initialized to 0.

```
    1  public static double f(double a, double b) {
→   2    double sum;
    3    sum = a + b;
    4    return sum;
    5  }
    6
    7  public static void main(String[] arguments) {
    8    double dOne = 1.5;
    9    double dTwo = 3.1;
   10    double total = 0;
   11    for (int i = 0; i <= 3; i++) {
   12      if (i < dOne) {
   13        total = total + f(dOne, dTwo);
   14      } else {
   15        total = total + f(total, dOne);
   16      }
   17    }
   18    System.out.println("total=" + total);
   19  }
```

main()				
dOne	dTwo	total	i	Output
1.5	3.1	0		
			0	

f(double, double)		
a	b	sum
1.5	3.1	0

Next, we execute line 3 and set `sum` to 4.6

```
    1  public static double f(double a, double b) {
    2    double sum;
→   3    sum = a + b;
    4    return sum;
    5  }
    6
    7  public static void main(String[] arguments) {
    8    double dOne = 1.5;
    9    double dTwo = 3.1;
   10    double total = 0;
   11    for (int i = 0; i <= 3; i++) {
   12      if (i < dOne) {
   13        total = total + f(dOne, dTwo);
   14      } else {
   15        total = total + f(total, dOne);
   16      }
   17    }
   18    System.out.println("total=" + total);
   19  }
```

main()				
dOne	dTwo	total	i	Output
1.5	3.1	0		
			0	

f(double, double)		
a	b	sum
1.5	3.1	~~0~~
		4.6

The value of `sum` is returned to the `main` function.

We then continue the execution of line 13 and set `total` to 4.6 $(0 + 4.6)$.

```
    1  public static double f(double a, double b) {
    2    double sum;
    3    sum = a + b;
    4    return sum;
    5  }
    6
    7  public static void main(String[] arguments) {
    8    double dOne = 1.5;
    9    double dTwo = 3.1;
   10    double total = 0;
   11    for (int i = 0; i <= 3; i++) {
   12      if (i < dOne) {
→  13        total = total + f(dOne, dTwo);
   14      } else {
   15        total = total + f(total, dOne);
   16      }
   17    }
   18    System.out.println("total=" + total);
   19  }
```

main()				
dOne	dTwo	total	i	Output
1.5	3.1	~~0~~		
		4.6	0	

The process continue. We now go back to line 11, increment the value of `i` to 1 and verify the condition of the `for`-loop.

```
    1  public static double f(double a, double b) {
    2    double sum;
    3    sum = a + b;
    4    return sum;
    5  }
    6
    7  public static void main(String[] arguments) {
    8    double dOne = 1.5;
    9    double dTwo = 3.1;
   10    double total = 0;
→  11    for (int i = 0; i <= 3; i++) {
   12      if (i < dOne) {
   13        total = total + f(dOne, dTwo);
   14      } else {
   15        total = total + f(total, dOne);
   16      }
   17    }
   18    System.out.println("total=" + total);
   19  }
```

main()				
dOne	dTwo	total	i	Output
1.5	3.1	~~0~~		
		4.6	~~0~~	
			1	

The cycle repeat and in the next iteration, the values of `total` and `i` are updated to 9.2 and 2, respectively.

```
    1  public static double f(double a, double b) {
    2    double sum;
    3    sum = a + b;
    4    return sum;
    5  }
    6
    7  public static void main(String[] arguments) {
    8    double dOne = 1.5;
    9    double dTwo = 3.1;
   10    double total = 0;
→  11    for (int i = 0; i <= 3; i++) {
   12      if (i < dOne) {
   13        total = total + f(dOne, dTwo);
   14      } else {
   15        total = total + f(total, dOne);
   16      }
   17    }
   18    System.out.println("total=" + total);
   19  }
```

main()				
dOne	dTwo	total	i	Output
1.5	3.1	~~0~~		
		~~4.6~~	~~0~~	
		9.2	~~1~~	
			2	

In the next iteration, as `i` is now greater than `dOne`, line 15 will be executed, passing `total` and `dOne` as arguments to the function `f`.

```
    1  public static double f(double a, double b) {
→   2    double sum;
    3    sum = a + b;
    4    return sum;
    5  }
    6
    7  public static void main(String[] arguments) {
    8    double dOne = 1.5;
    9    double dTwo = 3.1;
   10    double total = 0;
   11    for (int i = 0; i <= 3; i++) {
   12      if (i < dOne) {
   13        total = total + f(dOne, dTwo);
   14      } else {
   15        total = total + f(total, dOne);
   16      }
   17    }
   18    System.out.println("total=" + total);
   19  }
```

main()				
dOne	dTwo	total	i	Output
1.5	3.1	~~0~~		
		~~4.6~~	~~0~~	
		9.2	~~1~~	
			2	

f(double, double)		
a	b	sum
9.2	1.5	0

After executed lines 3 and 4, the value of `sum` is then set to 10.7 and return back to the `main` function in line 15.

Hence, the value of `total` becomes 19.9 $(9.2 + \texttt{f}(9.2, 1.5))$.

```
   1  public static double f(double a, double b) {
   2    double sum;
   3    sum = a + b;
   4    return sum;
   5  }
   6
   7  public static void main(String[] arguments) {
   8    double dOne = 1.5;
   9    double dTwo = 3.1;
  10    double total = 0;
  11    for (int i = 0; i <= 3; i++) {
  12      if (i < dOne) {
  13        total = total + f(dOne, dTwo);
  14      } else {
→ 15        total = total + f(total, dOne);
  16      }
  17    }
  18    System.out.println("total=" + total);
  19  }
```

main()				
dOne	dTwo	total	i	Output
1.5	3.1	~~0~~		
		~~4.6~~	~~0~~	
		~~9.2~~	~~1~~	
		19.9	2	

f(double, double)		
a	b	sum
9.2	1.5	~~0~~
		10.7

The cycle repeat as `i` become 3 in the next iteration. So the values of `total` and `i` becomes 41.3 $(19.9 + \texttt{f}(19.9, 1.5))$ and 4, respectively.

```
   1  public static double f(double a, double b) {
   2    double sum;
   3    sum = a + b;
   4    return sum;
   5  }
   6
   7  public static void main(String[] arguments) {
   8    double dOne = 1.5;
   9    double dTwo = 3.1;
  10    double total = 0;
→ 11    for (int i = 0; i <= 3; i++) {
  12      if (i < dOne) {
  13        total = total + f(dOne, dTwo);
  14      } else {
  15        total = total + f(total, dOne);
  16      }
  17    }
  18    System.out.println("total=" + total);
  19  }
```

main()				
dOne	dTwo	total	i	Output
1.5	3.1	~~0~~		
		~~4.6~~	~~0~~	
		~~9.2~~	~~1~~	
		~~19.9~~	~~2~~	
		41.3	~~3~~	
			4	

As i $= 4$, the condition in the loop no longer valid. Therefore, we continue the execution by running line 18 and output the value of total to the output.

```
     1  public static double f(double a, double b) {
     2    double sum;
     3    sum = a + b;
     4    return sum;
     5  }
     6
     7  public static void main(String[] arguments) {
     8    double dOne = 1.5;
     9    double dTwo = 3.1;
    10    double total = 0;
    11    for (int i = 0; i <= 3; i++) {
    12      if (i < dOne) {
    13        total = total + f(dOne, dTwo);
    14      } else {
    15        total = total + f(total, dOne);
    16      }
    17    }
→   18    System.out.println("total=" + total);
    19  }
```

main()				
dOne	dTwo	total	i	Output
1.5	3.1	~~0~~		
		~~4.6~~	~~0~~	
		~~9.2~~	~~1~~	
		~~19.9~~	~~2~~	
		41.3	~~3~~	
			4	41.3

As can be seen, code tracing does not just help you understand the code that works correctly. It is a powerful technique for finding errors in your code.

When a program behaves in a way that you don't expect, you should get out a sheet of paper and track the values of the variables as you mentally step through the code.

> You don't need a working program to do code tracing! You can do code tracing with pseudocode. In fact, it has always been a good practice to hand-trace your pseudocode before going into the actual coding. Doing this can confirm that your algorithm works correctly.

Practical Exercises

Problem 1. What values will variables a and b have at the end of the program?

```
public static void main(String[] arguments) {
  int a;
  a = 5;
  int b;
  b = 3;
  int c;
  c = a;
  a = b;
  b = c;
}
```

Problem 2. Trace the following code segment and find the value of the last value printed.

```
int first = 1;
int second = 1;
while (second <= 10)
{
  System.out.println(second);
  int temp = first + second;
  first = second;
  second = temp;
}
```

Problem 3. Trace the snippet below, and shows all outputs.

```
int i = 0;
int t = 0;
while (i < 5)
{
  i++;
  System.out.println(i + ": " + (i * 2));
  t += i;
}
System.out.println("t is " + t);
```

Problem 4. Trace the snippet below and shows the output at the end.

```
int[] nums = { 1, 2, 4, 6, 7, 3, 8, 5 };
boolean sorted = true;
for (int i = 0; i < nums.length; i++)
{
  if (nums[i] > nums[i + 1])
  {
    sorted = false;
    break;
  }
}
System.out.println("sorted=" + sorted);
```

Problem 5. What is the output of the following code snippet?

```
int x = 0;
int y = 0;
while (x < 123)
{
  x = x + 2;
  y = y + 1;
}
System.out.println(y);
```

Problem 6. What is the output of the following code snippet?

```
int i = 0;
while (i != 5)
{
  System.out.print(i + " ");
  i++;
  if (i == 5) System.out.println("end");
}
```

Problem 7. What is the output of the following code snippet?

```
int n = 1;
while (n < 100)
{
  n = 2 * n;
  System.out.print(n + " ");
}
```

Problem 8. Trace the following code snippet code and find the value(s) printed.

```
$s =$"Fred"
$r =$""
$i = 0$
**while** $i <$ length of $s$
    $c =$ i$^{\text{th}}$ character of $s$
    $r = c + r$
    $i++$
Print $r$
```

Problem 9. Trace the snippet below and shows all outputs.

```java
public class MyClass {

  private static int a = 1;

  public static void main(String[] arguments) {
    {
      double a;
      a = 6;
      System.out.println(a);
    }
    System.out.println(a);
  }

}
```

Problem 10. What is the output of the following code snippet?

```java
int month = 0;
double principal = 100;
double interest = 5;
while (month > 5) {
  principal = (principal * (100 + interest)) / 100;
  System.out.println("principal=" + principal);
}
```

A. The code snippet will display the interest plus the principal calculated for five months.

B. The code snippet will continue to display the calculated interest forever because the loop not end

C. The code snippet will not display any output because the loop condition was not satisfied

D. The code snippet will not display any output because it will *not* compile

Problem 11. What is the output of the following code snippet?

```java
int month = 0;
double principal = 100;
double interest = 5;
while (month < 5) {
  principal = (principal * (100 + interest)) / 100;
  System.out.println("principal=" + principal);
}
```

A. The code snippet will display the interest plus the principal calculated for five months.

B. The code snippet will continue to display the calculated interest forever because the loop not end

C. The code snippet will not display any output because the loop condition was not satisfied

D. The code snippet will not display any output because it will *not* compile

Problem 12.

(a) Trace the following code segment and find the last value printed.

```
int n = 1;
while (n <= 3)
{
  int r = 2 * n * n;
  System.out.print(r + ", ");
  n++;
}
System.out.println();
```

(b) The snippet above shows a potential error. Usually, commas are between values only. However, as can be seen from the trace, there is a comma after the last value.

Rearrange the code below to produce a loop that does not have this problem.

```
int n = 1;
while (n <= 3)
{
if (n > 1) System.out.print(", ");
int r = 2 * n * n;
System.out.print(r);
n++;
}
System.out.println();
```

Problem 13. Trace the snippet below and shows the output at the end.

```
private static int f(int n) {
  if (n == 0)
  {
    return 0;
  }
  else
  {
    if (n == 1)
    {
      return 1;
    }
    else
    {
      return f(n - 1) + f(n - 2);
    }
  }
}

public static void main(String[] arguments) {
  int x = f(6);
  System.out.println(x);
}
```

Problem 14. The following pseudocode is intended to count the number of digits in the positive integer n.

```
count = 1
temp = n
while temp > 10
    increment count
    divide temp by 10.0
```

Trace the pseudocode for (i) $n = 123$, (ii) $n = 100$; and (iii) $n = 3$.

What errors do you find, and how you fix the code?

A. The code is wrong for all inputs. The loop condition should be `temp!=10`.

B. The code is wrong for inputs that are divisible by ten. The loop condition should be `temp>=10`.

C. The code is wrong for inputs that are less than ten. The loop condition should be `temp>0`.

D. There is no error in the code.

Problem 15. Trace the snippet below and shows the output.

```java
private static int f(int a, int b) {
  if (a > b) return a;
  return b;
}

public static void main(String[] arguments) {
  int x = 3;
  int y = 9;
  int v = f(x, y);
  System.out.println(v);
}
```

Solutions

1. a=3, b=5
2. 8
3. 1: 2
   2: 4
   3: 6
   4: 8
   5: 10
   t is 15
4. 8
5. 62
6. 01234end
7. 2 4 8 16 32 64 128
8. derF
9. 6.0
   1
10. C
11. B
12. (a) 2, 8, 18,

    (b)

```
int n = 1;
while (n <= 3)
{
  int r = 2 * n * n;
  System.out.print(r);
  if (n > 1) System.out.print(", ");
  n++;
}
System.out.println();
```

13. 8
14. B
15. 9