This document contains two sets of exercises dealing with for-loops that process a range m..n of integers. When doing them, do not worry about declaring variables; assume all variables are declared.

These exercises were written early in the morning and were not proofread by anyone. Further, the code was not checked in DrJava. Hence, there may be mistakes. We apologize for them in advance.

A. In each of the problems below,

1. Write an assertion at the end, the "postcondition", that indicates what is true at the end.
2. Fill in the first line of the loop, with initialization, loop condition, and increment.

Note that we are not asking you to complete the loop, but just to gain practice in the two tasks shown above. Answers are given at the end.

A1. // Store in x the sum of 3..n

```
for (int k=                    ) {
    // Process k
}
//
```

A2. // Set x to the number of 'e's in String s

```
for (int k=                ) {
    // Process k
}
//
```

A3. // Set b to the value of the assertion
// "no int in 2..n-1 divides n"

```
for (int k=                        ) {
    // Process k
}
//
```

A4. // Assume s1 and s2 have the same length
// Given are two string s1 and s2
// Set b to the value of the assertion
// "s2 is the reverse of s1"

```
for (int k=                        ) {
    // Process k
}
//
```

A5. **int** n= s.length() – 1;
// Set b to the value of the sentence,
// "String s is a palindrome", i.e.
// "s[0..n/2 – 1] is the reverse of s[n – n/2.. n–1]"

```
for (int k=                        ) {
    // Process k
}
//
```

A6. For this problem, note that the number of days in each month of the year is: Jan, Mar, May, Jul, Aug, Oct, Dec: 31, Apr, June, Sep, Nov: 30. Feb: 28. Therefore, the number of days before month 3 (March) is 31 + 28 = 59.

```
// 1 ≤ m ≤ 12: so m is the number of a month.
// Set n to the no. of days in months 1..m–1.
for (int k=                        ) {
    // Process k
}
//
```

A7. // Set v to the sum
// 1*(n–1) + 2*(n–2) + … + (n-1)*(n – (n–1))

```
for (int k=                        ) {
    // Process k
}
//
```

A8. Assume that a function isPrime(n) yields true if n is a prime and false otherwise.

```
// Print all primes in the range –15..15
for (int k=                        ) {
    // Process k
}
//
```

A9. // Set v to the sum: $1/1^2 + 1/2^2 + 1/3^2 + \ldots + 1/n^2$.

```
for (int k=                        ) {
    // Process k
}
//
```

B. In each of the problems shown below, we give you the specification as a command, the "postcondition", an assertion that says what is true after execution of the code, and the straightforward beginning of the loop to process the range. You should

1. Write the invariant.
2. Fill in any necessary initialization of the variables to truthify the invariant.
3. Write the body of the loop to process integer k.

B1. // Store in x the sum of 3..n

```
// invariant:
for (int k= 3; k <= n; k= k+1) {
    // Process k
```

```
}
// x is the sum of 3..n
```

B2.
```
// Set x to the number of 'e's in String s

// invariant:
for (int k= 0; k < s.length(); k= k+1) {
    // Process k



}
// x = no. of 'e's in s[0..s.length()-1]
```

B3.
```
// Set b to the value of the assertion
// "no int in 2..n-1 divides n"


// invariant:
for (int k= 2; k < n; k= k+1) {
    // Process k



}
// b = "no int in 2..n-1 divides n"
```

B4.
```
// Assume s1 and s2 have the same length
// Set b to the value of the assertion
// "s2 is the reverse of s1"

// invariant:

for (int k=  0; k < s1.length() ; k= k+1) {
    // Process k



}
// s1(0..s.length()–1) = reverse of s2[0..s.length()–1)
```

B5.
```
int n= s.length() – 1;
 // Set b to the value of the sentence,
//  "String s is a palindrome", i.e.
//  "s[0..n/2 – 1] is the reverse of s[n – n/2.. n–1]"


// invariant:

for (int k=  0; k <= n/2 – 1; k= k+1) {
    // Process k



}
// b = "s[0..n/2 – 1] is the reverse of s[n – n/2.. n–1]"
```

B6.
```
// 1 ≤ m ≤ 12: so m is the number of a month.
// Set n to the no. of days in months m-1.

// invariant:

for (int k=   1; k < m; k= k+1) {
    // Process k



}
// n =  no. of days in month 1..m-1.
```

B7.
```
// Set v to the sum
// 1*(n–1) + 2*(n–2) + … + (n-1)*(n – (n–1))


// invariant:

for (int k= 1; k < n; k= k+1) {
    // Process k



}
// v = 1*(n–1) + 2*(n–2) + … + (n-1)*(n – (n–1))
```

B8. Assume that a function isPrime(n) yields true if n is a prime and false otherwise.
```
// Print all primes in the range –15..15


// invariant:
for (int k=  –15; k <= 15; k= k+1) {
    // Process k



}
// All primes in range –15..15 have been printed.
```

B9.
```
// Set v to the sum: 1/1^2 + 1/2^2+ 1/3^2 + … + 1/n^2.


// invariant

for (int k=  1; k <= n; k= k+1) {
```

```
        // Process k




    }
    // v =1/1^2 + 1/2^2+ 1/3^2 + … + 1/n^2
```

Answers

A1.
```
// Store in x the sum of 3..n
for (int k= 3; k <= n; k= k+1) {
    // Process k

}
// x = the sum of 3..n
```

A2.
```
// Set x to the number of "e"s in String s
for (int k= 0; k < s.length(); k= k+1) {
    // Process k
}
// x = no. of "e"s in s[0..s.length()–1
```

A3.
```
// Set b to the value of the assertion
// "no int in 2..n-1 divides n"
for (int k= 2; k < n; k= k+1) {
    // Process k
}
// b = "no int in 2..n-1 divides n"
```

A4.
```
// Assume s1 and s2 have the same length
// Set b to the value of the assertion
// "s2 is the reverse of s1"
for (int k= 0; k < s1.length() ; k= k+1) {
    // Process k
}
// s1(0..s.length()–1) = reverse of s2[0..s.length()–1)
```

A5.
```
int n= s.length() – 1;
 // Set b to the value of the sentence,
//  "String s is a palindrome", i.e.
//  "s[0..n/2 – 1] is the reverse of s[n – n/2.. n–1]"
// inv: b = "s[0..k – 1] is the reverse of s[n – k.. n–
1]"
for (int k= 0; k <= n/2 – 1; k= k+1) {
    // Process k
}
// b = "s[0..n/2 – 1] is the reverse of s[n – n/2.. n–1]"
```

A6.
```
// 1 ≤ m ≤ 12: so m is the number of a month.
// Set n to the no. of days in months m-1.
for (int k= 1; k < m; k= k+1) {
    // Process k
}
// n = no. of days in month 1..m-1.
```

A7.
```
// Set v to the sum
// 1*(n–1) + 2*(n–2) + … + (n-1)*(n – (n–1))
for (int k= 1; k < n; k= k+1) {
    // Process k
}
// v = 1*(n–1) + 2*(n–2) + … + (n-1)*(n – (n–1))
```

A8. Assume that a function isPrime(n) yields true if n is a prime and false otherwise.
```
// Print all primes in the range –15..15
for (int k= –15; k <= 15; k= k+1) {
    // Process k
}
// All primes in range –15..15 have been printed.
```

A9.
```
// Set v to the sum: 1/1^2 + 1/2^2+ 1/3^2 + … + 1/n^2.
for (int k= 1; k <= n; k= k+1) {
    // Process k
}
// v =1/1^2 + 1/2^2+ 1/3^2 + … + 1/n^2
```

B1.
```
// Store in x the sum of 3..n
x= 0;
// invariant: x is the sum of 3..k–1
for (int k= 3; k <= n; k= k+1) {
    x= x + k;
}
// x is the sum of 3..n
```

B2.
```
// Set x to the number of "e"s in String s
x= 0;
// invariant: x = no. of "e"s in s[0..k-1]
for (int k= 0; k < s.length(); k= k+1) {
    if (s.charAt(k) == 'e') {
      x= x + e;
    }
}
// x = no. of "e"s in s[0..s.length()-1]
```

B3.
```
// Set b to the value of the assertion
// "no int in 2..n-1 divides n"
b= true;
// invariant: b = "no int in 2..k-1 divides n"
for (int k= 2; k < n; k= k+1) {
    if (n % k == 0) {
       b= false;
    }
}
// b = "no int in 2..n-1 divides n"
```

B4.
```
// Assume s1 and s2 have the same length
// Set b to the value of the assertion
// "s2 is the reverse of s1"
b= true;
// invariant: s1(0..k–1) =
//    reverse of s2[s.length()-k..s.length()–1)
```

```
for (int k=  0; k < s1.length() ; k= k+1) {
    if  (s1.charAt(k) != s2.charAt(s.length–k–1)) {
        b= false;
    }
}
// s1(0..s.length()–1) = reverse of s2[0..s.length()–1)
```

B5.
```
int n= s.length() – 1;
 // Set b to the value of the sentence,
//  "String s is a palindrome", i.e.
//  "s[0..n/2 – 1] is the reverse of s[n – n/2.. n–1]"
b= true;
// invariant: b = "s[0..k–1] is the reverse of
                    s[n–k.. n–1]"
for (int k=  0; k <= n/2 – 1; k= k+1) {
    if (s.charAt(k) != s.charAt(n–k–1) {
      b= false;
    }
}
// b = "s[0..n/2 – 1] is the reverse of s[n – n/2.. n–1]"
```

B6.
```
// 1 ≤ m ≤ 12: so m is the number of a month.
// Set n to the no. of days in months m-1.
n= 0;
// invariant: n =  no. of days in month 1..k-1
for (int k=   1; k < m; k= k+1) {
    // Add the number of days in month m to n
    if (m == 28)
      n= n + 28;
    else if (m == 4 || m == 6 || m == 9 || m == 11) {
    }
    else n= n + 31;
}
// n =  no. of days in month 1..m-1.
```

B7.
```
// Set v to the sum
// 1*(n–1) + 2*(n–2) + … + (n-1)*(n – (n–1))
v= 0;
// invariant:
//   v = 1*(n–1) + 2*(n–2) + … + (n-1)*(n – (n–1))
for (int k= 1; k < n; k= k+1) {
    v= v + k*(n–k);
}
// v = 1*(n–1) + 2*(n–2) + … + (n-1)*(n – (n–1))
```

B8. Assume that a function isPrime(n) yields true if n is a prime and false otherwise.
```
// Print all primes in the range –15..15
// invariant: all primes in –15..k–1 have been printed
for (int k=  –15; k <= 15; k= k+1) {
    if (isPrime(k) {
      System.out.println(k);
    }
}
// All primes in range –15..15 have been printed.
```

B9.
```
// Set v to the sum: 1/1^2 + 1/2^2+ 1/3^2 + … + 1/n^2.
v= 0;
// invariant: v =1/1^2 + 1/2^2+ 1/3^2 + … + 1/(k–1)^2
for (int k=  1; k <= n; k= k+1) {
    v= v + 1/ (k*k);
}
// v =1/1^2 + 1/2^2+ 1/3^2 + … + 1/n^2
```