CS 61BL Summer 2021

Loops and Arrays

Quiz 2: Friday June 25, 2021

1 Looping

What output is produced by the following non standard for loop? Record it **exactly**. We strongly suggest that you use a table to keep track of the value of k.

```
for (int k = 1; k <= 10; k++) {
    k = k + 1;
    System.out.print(k + " ``);
}
```

Output:

2 While to For

Translate each of the following two while loops into for loops.

Note: You must get both parts correct to get credit for this question.

Part One

```
int k = 0;
while (k < 10) {
        System.out.println(k);
        k = k + 1;
}
```

Select the letter corresponding to the for loop below that is equivalent to the while loop above.

A

```
for (int i = 0; i < 10; i++) {
        System.out.println(i);
}
```

B

```
for (int i = 0; i < 10; i++) {
        i = i + 1;
        System.out.println(i);
}
```

C

```
for (int i = 1; i < 10; i++) {
        System.out.println(i);
}
```

D

```
for (int i = 1; i < 10; i++) {
        i = i + 1;
        System.out.println(i);
}
```

( ) A
( ) B
( ) C
( ) D

Part Two

```
int k = 0;
while (k < 10) {
        k = k + 1;
        System.out.println(k);
}
```

Select the letter corresponding to the for loop below that is equivalent to the above while loop.

A

```
for (int i = 0; i <= 10; i++) {
        System.out.println(i);
}
```

B

```
for (int i = 0; i <= 10; i++) {
        i = i + 1;
        System.out.println(i);
}
```

C

```
for (int i = 1; i <= 10; i++) {
        System.out.println(i);
}
```

D

```
for (int i = 1; i <= 10; i++) {
        i = i + 1;
        System.out.println(i);
}
```

( ) A
( ) B
( ) C
( ) D

3 Arrays

Refer to the following classes:

```
1  public class Point {
2          public int x;
3          public int y;
4          //implementation
5  }
6  public class Line {
7          public Point left_endpoint;
8          public Point right_endpoint;
9          public int slope;
10
11         public Line(Point one, Point two) {
12                 this.left_endpoint = one;
13                 this.right_endpoint = two;
14         }
15         //implementation
16 }
```

What is printed by the program below? Record it **exactly**.

```
1  public class Test {
2          public static void main(String [] ars) {
3                  Line[] line = new Line[2];
4                  Point p = new Point();
5                  Point q = new Point();
6                  Line pq = new Line(p, q);
7                  line[0] = pq;
8                  line[0].left_endpoint.x = 1;
9                  pq = new Line(q, p);
10                 line[1] = pq;
11                 line[0].left_endpoint.y = 2;
12                 line[1] = line[0];
13                 line[0] = pq;
14                 line[0].left_endpoint.x = 2;
15                 line[0].left_endpoint.y = 1;
16                 System.out.print(line[0].right_endpoint.x + " ");
17                 System.out.print(line[0].left_endpoint.x + " ");
18                 System.out.print(line[0].right_endpoint.y + " ");
19                 System.out.print(line[0].left_endpoint.y);
20         }
21 }
```

Output:

4 Build Code

Part One

Note: You must get both parts correct to get credit for this subpart.

You are writing a program `boolean isPrime(int n)` which is designed to determine if a number is prime. You know that the definition of a prime number is one that is divisible only by 1 and itself. Assume that `n` is a positive number and is greater than 1 throughout this entire question.

Which of the following boolean conditions is true for any number `1 < x < n` if the number `n` is prime. ( ) `n % x == 1`
( ) `n % x == x`
( ) `n % x != 0`
( ) None of the above

Which of the following would provide an inefficient but accurate implementation of the `boolean isPrime(int n)` method. Note that at least one of the below is correct.

Select the letter corresponding to the correct answer(s).

A

```
int x = 2;
while (x < n) {
        if (!condition) {
                return false;
        }
}
return true;
```

B

```
for (int x = 2; x < n; x++) {
        if (!condition) {
                return false;
        }
}
return true;
```

C

```
for (int x = 2; x < n; x++) {
        if (condition) {
                return true;
        }
}
return false;
```

D

```
for (int x = 2; x < n; x++) {
        if (!condition) {
```

```
3                 return true;
4         } else {
5                 return false;
6         }
7 }
```

[ ] A
[ ] B
[ ] C
[ ] D

Part Two

All of the implementations above are a tad inefficient, so let's make them slightly more efficient! To do so, we will change the loop's **stopping condition**. The stopping condition for *all* the implementations above is `x < n`. Which of the following stopping conditions are *correct* and would make the code *more efficient*?

Hint: You only need to check the factors until the square root of `n`

[ ] `x <= n * n`
[ ] `x * x <= n`
[ ] `x * x * x <= n`