Question 1

Examine the following code snippet:

```
int[] arr = {4, 2, 9, 7, 3};
for (int k : arr) {
    k = k + 10;
    System.out.print(k + " ");
}
for (int k : arr)
    System.out.print(k + " ");
```

What is the output when this code is executed?

(A) 0 1 2 3 4 0 1 2 3 4

(B) 4 2 9 7 3 4 2 9 7 3

(C) 14 12 19 17 13 0 1 2 3 4

(D) 14 12 19 17 13 4 2 9 7 3

(E) 14 12 19 17 13 14 12 19 17 13

Question 2

Consider the following method:

```
public int mystery(int num) {
    int x = num;
    while (x > 0) {
        if (x / 10 % 2 == 0)
            return x;
        x = x / 10;
    }
    return x;
}
```

What value is returned by the call mystery(1034)?

(A) 4

(B) 10

(C) 34

(D) 103

(E) 1034

Question 3

Study the following method:

```
public int pick(boolean test, int x, int y) {
    if (test)
        return x;
    else
        return y;
}
```

What is the result of the method call pick(false, pick(true, 0, 1), pick(true, 6, 7))?

(A) 0

(B) 1

(C) 3

(D) 6

(E) 7

Question 4

Analyze the following class hierarchy and code:

```
public class First {
    public String name() {
        return "First";
    }
}
public class Second extends First {
    public void whoRules() {
```

```
        System.out.print(super.name() + " rules");
        System.out.println(" but " + name() + " is even better");
    }
    @Override
    public String name() {
        return "Second";
    }
}
public class Third extends Second {
    @Override
    public String name() {
        return "Third";
    }
}
```

```
Second varSecond = new Second();
Third varThird = new Third();
varSecond.whoRules();
varThird.whoRules();
```

What is printed when this code runs?

(A) First rules but Second is even betterFirst rules but Second is even better

(B) First rules but Second is even betterFirst rules but Third is even better

(C) First rules but Second is even betterSecond rules but Second is even better

(D) First rules but Second is even betterSecond rules but Third is even better

(E) Second rules but Second is even betterSecond rules but Second is even better

Question 5

Consider the code below:

```
/* SomeType1 */ varA = new Second();
/* SomeType2 */ varB = new Third();
```

```
varA.whoRules();
varB.whoRules();
```

Which type declarations for SomeType1 and SomeType2 will allow the code to compile without errors?I. First and ThirdII. Second and SecondIII. Third and Third

(A) I only

(B) II only

(C) III only

(D) I and II

(E) II and III

Question 6

Review the following class definitions and statement:

```
public class Base {
    public Base() {
        System.out.print("Base ");
    }
}
public class Derived extends Base {
    public Derived() {
        System.out.print("Derived ");
    }
}
```

```
Derived d1 = new Derived();
```

What is printed when the object is created?

(A) No output, as it is a variable declaration.

(B) Base

(C) Derived

(D) Base Derived

(E) Derived Base

Question 7

The following method merges two queues:

```
public Queue<Integer> merge(Queue<Integer> a, Queue<Integer> b) {
    Queue<Integer> result = new LinkedList<>();
    while (!a.isEmpty() || !b.isEmpty()) {
        if (/* missing condition */)
            result.add(a.remove());
        else
            result.add(b.remove());
    }
    return result;
}
```

Which condition ensures the merge works correctly (maintaining non-decreasing order)?

(A) !a.isEmpty() && b.isEmpty()

(B) a.peek().intValue() < b.peek().intValue()

(C) b.isEmpty() || (a.peek().intValue() < b.peek().intValue())

(D) !b.isEmpty() && (a.peek().intValue() < b.peek().intValue())

(E) b.isEmpty() || (!a.isEmpty() && (a.peek().intValue() < b.peek().intValue()))

Question 8

Examine the following method:

```
public int getTheResult(int n) {
    int product = 1;
    for (int number = 1; number < n; number++) {
        if (number % 2 == 0)
            product *= number;
    }
```

```
    return product;
}
```

What is the return value of getTheResult(8)?

(A) 48

(B) 105

(C) 384

(D) 5040

(E) 40320

Question 9

Consider the code that transforms an array:

```
int[] oldArray = {1, 2, 3, 4, 5, 6, 7, 8, 9};
int[][] newArray = new int[3][3];
int row = 0, col = 0;
for (int index = 0; index < oldArray.length; index++) {
    newArray[row][col] = oldArray[index];
    row++;
    if (row % 3 == 0) {
        col++;
        row = 0;
    }
}
System.out.println(newArray[0][2]);
```

What is the output of this code?

(A) 3

(B) 4

(C) 5

(D) 7

(E) 8

Question 10

Analyze the ArrayList operations:

```
ArrayList<Integer> nums = new ArrayList<>();
nums.add(37);
nums.add(0);
nums.add(3);
nums.add(1, 2);
nums.set(0, 1);
nums.remove(2);
System.out.println(nums);
```

What is printed after executing these statements?

(A) [1, 2, 0]

(B) [1, 3, 0]

(C) [1, 3, 2]

(D) [1, 37, 3, 0]

(E) [37, 0, 0]

Question 11

Given boolean variables a, b, c, d, which expression is equivalent to !(!(a && b) || (c || !d))?

(A) (a && b) && (!c && d)

(B) (a || b) && (!c && d)

(C) (a && b) || (c || !d)

(D) (!a || !b) && (!c && d)

(E) !(a && b) && (c || !d)

Question 12

Consider the class hierarchy:

```
public class A {
    private int x;
```

```
    public A() { x = 0; }
    public A(int y) { x = y; }
}
public class B extends A {
    private int y;
    public B() {
        /* missing code */
    }
}
```

Which code initializes both x and y to 0 when new B() is called?I. y = 0;II. super(0); y = 0;III. x = 0; y = 0;

(A) I only

(B) II only

(C) I and II only

(D) II and III only

(E) I, II, and III

Question 13

Study the recursive method:

```
public int surprise(int b) {
    if (b % 2 == 0) {
        if (b < 10)
            return b;
        else
            return (b % 10) + surprise(b / 10);
    } else {
        if (b < 10)
            return 0;
        else
            return surprise(b / 10);
```

```
    }
}
```

Which of the following is true?I. surprise(146781) == 0II. surprise(7754) == 4III. surprise(58216) == 16

(A) I only

(B) II only

(C) III only

(D) II and III only

(E) I, II, and III

Question 14

The goal is to check if an array is in strictly decreasing order. Which loop correctly implements this?I.

```
for (int k = 0; k < nums.length; k++)
    if (nums[k] <= nums[k+1])
        return false;
return true;
```

II.

```
for (int k = 1; k < nums.length; k++)
    if (nums[k-1] <= nums[k])
        return false;
return true;
```

III.

```
for (int k = 0; k < nums.length - 1; k++) {
    if (nums[k] <= nums[k+1])
        return false;
    else
        return true;
```

```
}
return true;
```

(A) I only

(B) II only

(C) III only

(D) I and III

(E) II and III

Question 15

Given a binary tree referenced by TreeNode t, the method funny is defined as:

```
public void funny(TreeNode p) {
    if (p.getLeft() != null)
        funny(p.getLeft());
    else if (p.getRight() != null)
        funny(p.getRight());
    else
        System.out.print(p.getValue() + ".");
}
```

What does funny(t) print?

(A) how.now.brown.cow.this.is.only.a.test.

(B) how.now.cow.brown.this.only.a.is.test.

(C) cow.only.a.test.

(D) cow.

(E) No output due to a recursion error.

Question 16

Which data structure best supports a web browser's "back" functionality, where the most recent site is returned first?

(A) Stack

(B) Queue

(C) PriorityQueue

(D) TreeNode

(E) TreeMap

Question 17

The TemperatureReading class implements Comparable. Which compareTo implementations correctly order by increasing temperature?I.

```
Double d1 = new Double(temperature);
Double d2 = new Double(other.temperature);
return d1.compareTo(d2);
```

II.

```
if (temperature < other.temperature)
    return -1;
else if (temperature == other.temperature)
    return 0;
else
    return 1;
```

III.

```
return (int)(temperature - other.temperature);
```

(A) II only

(B) I and II only

(C) I and III only

(D) II and III only

(E) I, II, and III

Question 18

Consider the recursive string manipulation method:

```
public String recScramble(String str, int[] positions, int k) {
    if (str == null || str.length() == 0)
        return "";
    if (str.length() == 1)
        return str;
    int pos = positions[k];
    String nstr = str.substring(pos, pos + 1);
    str = str.substring(0, pos) + str.substring(pos + 1);
    return nstr + recScramble(str, positions, k + 1);
}
```

What is the output of recScramble("epic", new int[] {2, 1, 1}, 0)?

(A) cepi

(B) epci

(C) iecp

(D) iepc

(E) ipce

Question 19

Analyze the stack operations:

```
Stack<Integer> stackOne = new Stack<>();
Stack<Integer> stackTwo = new Stack<>();
for (int k = 0; k < 5; k++)
    stackOne.push(k);
while (!stackOne.isEmpty())
    stackTwo.push(stackOne.pop());
for (int k = 5; k < 10; k++)
    stackTwo.push(k);
while (!stackTwo.isEmpty())
    System.out.print(stackTwo.pop() + " ");
```

What is printed?

(A) 0 1 2 3 4 5 6 7 8 9

(B) 0 1 2 3 4 9 8 7 6 5

(C) 5 6 7 8 9 0 1 2 3 4

(D) 9 8 7 6 5 0 1 2 3 4

(E) 9 8 7 6 5 4 3 2 1 0

Question 20

Consider the recursive function:

```
public int addFun(int n) {
    if (n <= 0)
        return 0;
    if (n == 1)
        return 2;
    return addFun(n - 1) + addFun(n - 2);
}
```

What is the result of addFun(6)?

(A) 10

(B) 12

(C) 16

(D) 26

(E) 32

Question 21

A CornerBug turns at 90-degree angles instead of 45 degrees. What is the best design approach?

(A) Extend Bug and override the turn method.

(B) Implement a RightTurn interface.

(C) Add a constructor parameter to Bug.

(D) Create an abstract RightAngleBug class.

(E) Define a new CornerBug interface.

Question 22

For LeftTurningBug (a subclass of Bug), which code turns the bug 90 degrees left?I. setDirection(Location.LEFT);II. setDirection(getDirection() + Location.LEFT);III. setDirection(Location.HALF_LEFT + Location.HALF_LEFT);

(A) I only

(B) II only

(C) III only

(D) I and III only

(E) I, II, and III

Question 23

The MysteryBug class:

```
public class MysteryBug extends Bug {
    private int count;
    public MysteryBug() {
        setDirection(Location.NORTHEAST);
        count = 0;
    }
    public void act() {
        if (count < 3 && canMove()) {
            move();
            count++;
        } else {
            turn();
            count = 0;
        }
    }
}
```

What pattern does this bug create?

(A) No pattern (no flowers placed).

(B) Square pattern (like BoxBug).

(C) Diamond pattern (rotated square).

(D) Six-sided pattern.

(E) Eight-sided pattern.

Question 24

A PouncingCritter moves to a random occupied location. Which methods need overriding?
I. getMoveLocationsII. makeMoveIII. selectMoveLocation

(A) I only

(B) II only

(C) III only

(D) I and III only

(E) I, II, and III

Question 25

The method to move an actor to a new grid:

```
public void moveActorToNewGrid(Actor anActor, Grid<Actor> newGrid) {
    Grid<Actor> oldGrid = anActor.getGrid();
    Location loc = anActor.getLocation();
    /* missing code */
}
```

Which code correctly moves the actor while maintaining grid consistency?(A) anActor.putSelfInGrid(newGrid, loc); anActor.removeSelfFromGrid();

(B) oldGrid.remove(loc); anActor.putSelfInGrid(newGrid, loc);

(C) anActor.removeSelfFromGrid(); anActor.putSelfInGrid(newGrid, loc);

(D) oldGrid.remove(loc); newGrid.put(anActor, loc);

(E) newGrid.put(anActor, loc); oldGrid.remove(loc);

Question 26

The fillQ method constructs a queue:

```
public static Queue<Integer> fillQ(int count) {
```

```
    Queue<Integer> intQ = new LinkedList<>();
    int x = 3, y = 2, t = 1;
    if (count == 1)
        intQ.add(y);
    else {
        intQ.add(y);
        intQ.add(t);
        for (int k = 2; k < count; k++) {
            x = y + t - k;
            y = t;
            t = x + k;
            intQ.add(t);
        }
    }
    return intQ;
}
```

What is the output of System.out.println(fillQ(6))?

(A) [1, 2, 3, 5, 8, 13]

(B) [2, 1, 3, 4, 7, 11]

(C) [2, 1, 3, 6, 10, 15]

(D) [2, 1, 3, 6, 12, 24]

(E) [2, 3, 5, 8, 12, 17]

Question 27

In the fillQ method, what does the value added at Point A represent?

(A) Sum of all previous queue elements.

(B) Sum of previous odd-valued elements.

(C) Sum of the two preceding elements.

(D) Sum of the loop index and the previous element.

(E) Square of the loop index.

Question 28

The binary tree method:

```
public int mystery(TreeNode t) {
    int keep = (Integer) t.getValue();
    int leftKeep = keep, rightKeep = keep;
    if (t.getLeft() != null)
        leftKeep = mystery(t.getLeft());
    if (t.getRight() != null)
        rightKeep = mystery(t.getRight());
    if (keep >= leftKeep) keep = leftKeep;
    if (keep >= rightKeep) keep = rightKeep;
    return keep;
}
```

What does this method return for a non-empty tree?(A) The smallest integer in the tree.(B) The largest integer in the tree.(C) The value of the root node.(D) The value of the rightmost leaf.(E) The value of the leftmost leaf.

Question 29

The ListNodeIterator class:

```
public class ListNodeIterator implements Iterator {
    private ListNode current;
    public ListNodeIterator(ListNode first) { current = first; }
    public Object next() {
        Object value = current.getValue();
        current = current.getNext();
        return value;
    }
    public boolean hasNext() {
        /* missing code */
    }
}
```

Which code correctly implements hasNext()?

(A) return current == null;

(B) return current != null;

(C) return current.getValue() != null;

(D) return current.getNext() != null;

(E) return current.getNext().getValue() != null;

Question 30

The removeName method aims to delete all occurrences of a name from a list. Which implementations work?I.

```
for (String name : nameList) {
    if (name.equals(nameToRemove))
        nameList.remove(name);
}
```

II.

```
for (int k = 0; k < nameList.size(); k++) {
    if (nameList.get(k).equals(nameToRemove))
        nameList.remove(k);
}
```

III.

```
for (int k = nameList.size() - 1; k >= 0; k--) {
    if (nameList.get(k).equals(nameToRemove))
        nameList.remove(k);
}
```

(A) I only

(B) II only

(C) III only

(D) II and III only

(E) I, II, and III

Question 31

Given the inorder traversal TCLSONAD and preorder traversal SCTLOAND of a binary tree, what is the right child of the root?

(A) A

(B) L

(C) N

(D) O

(E) S

Question 32

The linked list sum method:

```
public int sum(ListNode nums) {
    if (nums.getNext() == null)
        return 0;
    else
        return (Integer) nums.getValue() + sum(nums.getNext());
}
```

For which linked lists does this method return the correct sum?I. A list with one node.II. A list with two nodes.III. A list with three nodes.

(A) I only

(B) I and II only

(C) I and III only

(D) II and III only

(E) I, II, and III

Question 33

To print all entries in empMap (a TreeMap), which code works?I.

```
for (String key : empMap.keySet())
    System.out.println(key + " " + empMap.get(key));
```

II.

```
Set<String> keys = empMap.keySet();
Iterator<String> itr = keys.iterator();
while (itr.hasNext())
    System.out.println(keys + " " + itr.next());
```

III.

```
Set<String> keys = empMap.keySet();
Iterator<String> itr = keys.iterator();
while (itr.hasNext()) {
    String key = itr.next();
    System.out.println(key + " " + empMap.get(key));
}
```

(A) I only

(B) II only

(C) III only

(D) I and II

(E) I and III

Question 34

The boolean expression (a && (b || !a)) == (a && b) is true when:

(A) Only when a is true.

(B) Only when b is true.

(C) Only when both a and b are true.

(D) Never true.

(E) Always true.

Question 35

Mergesort takes 45 time units for 5,000 items. Approximate time for 20,000 items?

(A) 90

(B) 150

(C) 220

(D) 500

(E) 720

Question 36

Inserting TRULYBASIC into a binary search tree. What is the preorder traversal?(A) ABCILRSTUY(B) ACIBLSRYUT(C) TRLBAICSUY(D) TRULSYBAIC(E) YUSCIA BLRT

Question 37

After linking firstList and secondList with:

```
firstList.getNext().setNext(secondList.getNext());
secondList.setNext(firstList.getNext());
firstList.setNext(secondList);
secondList = null;
```

Which represents firstList?

(A) [1, 2]

(B) [1, 3, 4]

(C) [1, 3, 2, 4]

(D) [1, 3, 2, 3, 4]

(E) [1, 2, 3, 4]

Question 38

The PriorityQueue operations:

```
PriorityQueue<String> pq = new PriorityQueue<>();
pq.add("to"); pq.add("or"); pq.add("be");
pq.add("not"); pq.add("to"); pq.add("be");
while (!pq.isEmpty())
```

```
    System.out.print(pq.remove() + " ");
```

What is the output?

(A) to to or not be be

(B) to be or not to be

(C) be to not or be to

(D) be not or to

(E) be be not or to to

Question 39

The computeSum method's time complexity:

```
public int computeSum(int num) {
    int sum = 0, delta = 2;
    while (delta < 2 * num) {
        for (int k = delta / 2; k <= num; k += delta)
            sum += k;
        delta *= 2;
    }
    return sum;
}
```

(A) O(\log n)

(B) O((\log n)^2)

(C) O(n)

(D) O(n \log n)

(E) O(n^2)

Question 40

The TreeSet operations:

```
Set<String> cities = new TreeSet<>();
String[] arr1 = {"Philadelphia", "Minneapolis", "Houston"};
```

```
for (String s : arr1) cities.add(s);
String[] arr2 = {"Philadelphia", "Chicago"};
for (String s : arr2) cities.remove(s);
String[] arr3 = {"Des Moines", "Houston"};
for (String s : arr3) cities.add(s);
System.out.println(cities);
```

What is printed?

(A) [Chicago, Des Moines, Houston, Minneapolis, Philadelphia]

(B) [Des Moines, Houston, Minneapolis]

(C) [Des Moines, Houston, Minneapolis, Philadelphia]

(D) [Houston, Minneapolis]

(E) [Houston, Minneapolis, Philadelphia]