Questions 21-25 refer to the code from the GridWorld case study. A copy of the code is provided in the Appendix.

21. Consider the design of a `Grasshopper` class that extends `Bug`. When asked to move, a `Grasshopper` moves to a randomly chosen empty adjacent location that is within the grid. If there is no empty adjacent location that is within the grid, the `Grasshopper` does not move, but turns 45 degrees to the right without changing its location.

    Which method(s) of the `Bug` class should the `Grasshopper` class override so that a `Grasshopper` can behave as described above?

    I. `act()`

    II. `move()`

    III. `canMove()`

    (A) I only

    (B) II only

    (C) I and II only

    (D) II and III only

    (E) I, II, and III

22. Assume that `gus` has been defined and initialized as a `Bug` object in a class that contains the following code segment.

```
int numTurnsMade = 0;
for (int k = 1; k <= 100; k++)
{
  int dir = gus.getDirection();
  int dirTurn = dir + Location.HALF_RIGHT;
  gus.act();
  if ( /* expression */ )
    numTurnsMade++;
}
```

Which of the following could be used to replace `/* expression */` so that the variable `numTurnsMade` accurately stores the number of times that `gus` turns 45 degrees to the right?

(A) `dir == dirTurn`

(B) `dir == gus.getDirection()`

(C) `dirTurn == Location.HALF_RIGHT`

(D) `dirTurn == gus.getDirection()`

(E) `Location.HALF_RIGHT == gus.getDirection()`

GO ON TO THE NEXT PAGE.

23. Consider the following method that is intended to return an `ArrayList` of all the locations in `grd` that contain actors facing in direction `dir`.

```
public ArrayList<Location> findLocsFacingDir(int dir, Grid<Actor> grd)
{
  ArrayList<Location> desiredLocs = new ArrayList<Location>();

  for (Location loc : grd.getOccupiedLocations())
  {
    if ( /* expression */ == dir )
      desiredLocs.add(loc);
  }
  return desiredLocs;
}
```

Which of the following can be used to replace `/* expression */` so that `findLocsFacingDir` will work as intended?

(A) `loc.getDirection()`

(B) `getDirection(loc)`

(C) `((Actor) loc).getDirection()`

(D) `grd(loc).getDirection()`

(E) `grd.get(loc).getDirection()`

GO ON TO THE NEXT PAGE.

24. A `ColorChangingCritter` behaves like a `ChameleonCritter` but does not turn when it moves. A partial declaration for the `ColorChangingCritter` class is as follows.

```
public class ColorChangingCritter extends ChameleonCritter
{
  public void makeMove(Location loc)
  {  /* missing code */  }
}
```

Which of the following replacements for `/* missing code */` will correctly implement the desired behavior?

I. `moveTo(loc);`

II. `super.super.makeMove(loc);`

III.
```
int dir = getDirection();
super.makeMove(loc);
setDirection(dir);
```

(A) I only

(B) III only

(C) I and II only

(D) I and III only

(E) I, II, and III

GO ON TO THE NEXT PAGE.

25. A `MunchingCritter` acts by selecting one adjacent actor of any type, eating it (removing it from the grid), and moving to occupy its location. If there is no adjacent actor, the `MunchingCritter` moves like a normal critter. Consider the following three implementations of `MunchingCritter`.

Implementation I

```
public class MunchingCritter extends Critter
{
  private Location eatLoc;  // Remember location of critter that was eaten

  public void processActors(ArrayList<Actor> actors)
  {
    if (actors.size() == 0)
      eatLoc = null;
    else
    {
      Actor selected = actors.get(0);
      eatLoc = selected.getLocation();
      selected.removeSelfFromGrid();
    }
  }

  public Location selectMoveLocation(ArrayList<Location> locs)
  {
    if (eatLoc == null)
      return super.selectMoveLocation(locs);
    else
      return eatLoc;
  }
}
```

Implementation II

```
public class MunchingCritter extends Critter
{
  private Location eatLoc;  // Remember location of critter that was eaten

  public void processActors(ArrayList<Actor> actors)
  {
    if (actors.size() == 0)
      eatLoc = null;
    else
    {
      Actor selected = actors.get(0);
      eatLoc = selected.getLocation();
      selected.removeSelfFromGrid();
    }
  }

  public void makeMove(Location loc)
  {
    if (eatLoc == null)
      moveTo(loc);
    else
      moveTo(eatLoc);
  }
}
```

GO ON TO THE NEXT PAGE.

Implementation III

```
public class MunchingCritter extends Critter
{
  private boolean hasEaten;  // Remember if this critter ate something during this step

  public void processActors(ArrayList<Actor> actors)
  {
    if (actors.size() == 0)
      hasEaten = false;
    else
    {
      Actor selected = actors.get(0);
      Location moveLoc = selected.getLocation();
      selected.removeSelfFromGrid();
      moveTo(moveLoc);
      hasEaten = true;
    }
  }

  public void makeMove(Location loc)
  {
    if (!hasEaten)
      moveTo(loc);
  }
}
```

Which of the implementations would be considered to be well designed, in that they satisfy the postconditions in `Critter.java` ?

(A) I only

(B) II only

(C) III only

(D) I and II only

(E) I, II, and III

26. Assume that the array `arr` has been defined and initialized as follows.

```
int[] arr = /* initial values for the array */ ;
```

Which of the following will correctly print all of the odd integers contained in `arr` but none of the even integers contained in `arr` ?

(A)
```
for (int x : arr)
  if (x % 2 == 1)
    System.out.println(x);
```

(B)
```
for (int k = 1; k < arr.length; k++)
  if (arr[k] % 2 == 1)
    System.out.println(arr[k]);
```

(C)
```
for (int x : arr)
  if (x % 2 == 1)
    System.out.println(arr[x]);
```

(D)
```
for (int k = 0; k < arr.length; k++)
  if (arr[k] % 2 == 1)
    System.out.println(k);
```

(E)
```
for (int x : arr)
  if (arr[x] % 2 == 1)
    System.out.println(arr[x]);
```

Questions 27-28 refer to the following method.

```
public static int mystery(int n)
{
  int x = 1;
  int y = 1;

  // Point A

  while (n > 2)
  {
    x = x + y;

    // Point B

    y = x - y;
    n--;
  }

  // Point C

  return x;
}
```

27. What value is returned as a result of the call `mystery(6)` ?

(A) 1

(B) 5

(C) 6

(D) 8

(E) 13

---

28. Which of the following is true of method `mystery` ?

(A) `x` will sometimes be `1` at `// Point B`.

(B) `x` will never be `1` at `// Point C`.

(C) `n` will never be greater than `2` at `// Point A`.

(D) `n` will sometimes be greater than `2` at `// Point C`.

(E) `n` will always be greater than `2` at `// Point B`.

29. Consider the following code segment.

```
for (int k = 1; k <= 100; k++)
  if ((k % 4) == 0)
    System.out.println(k);
```

Which of the following code segments will produce the same output as the code segment above?

(A)
```
for (int k = 1; k <= 25; k++)
  System.out.println(k);
```

(B)
```
for (int k = 1; k <= 100; k = k + 4)
  System.out.println(k);
```

(C)
```
for (int k = 1; k <= 100; k++)
  System.out.println(k % 4);
```

(D)
```
for (int k = 4; k <= 25; k = 4 * k)
  System.out.println(k);
```

(E)
```
for (int k = 4; k <= 100; k = k + 4)
  System.out.println(k);
```

30. Consider the following method.

```
public static String scramble(String word, int howFar)
{
  return word.substring(howFar + 1, word.length()) +
         word.substring(0, howFar);
}
```

What value is returned as a result of the call `scramble("compiler", 3)`?

(A) `"compiler"`

(B) `"pilercom"`

(C) `"ilercom"`

(D) `"ilercomp"`

(E) No value is returned because an `IndexOutOfBoundsException` will be thrown.

GO ON TO THE NEXT PAGE.

31. Consider the following method.

```
public void mystery(int[] data)
{
  for (int k = 0; k < data.length - 1; k++)
    data[k + 1] = data[k] + data[k + 1];
}
```

The following code segment appears in another method in the same class.

```
int[] values = {5, 2, 1, 3, 8};
mystery(values);
for (int v : values)
  System.out.print(v + " ");
System.out.println();
```

What is printed as a result of executing the code segment?

(A) `5 2 1 3 8`

(B) `5 7 3 4 11`

(C) `5 7 8 11 19`

(D) `7 3 4 11 8`

(E) Nothing is printed because an `ArrayIndexOutOfBoundsException` is thrown during the execution of method `mystery`.

32. Consider the following method.

```
public int compute(int n, int k)
{
  int answer = 1;

  for (int i = 1; i <= k; i++)
    answer *= n;

  return answer;
}
```

Which of the following represents the value returned as a result of the call `compute(n, k)` ?

(A) $n*k$

(B) $n!$

(C) n^k

(D) 2^k

(E) k^n

GO ON TO THE NEXT PAGE.

33. Consider the following code segment.

```
int sum = 0;
int k = 1;
while (sum < 12 || k < 4)
  sum += k;

System.out.println(sum);
```

What is printed as a result of executing the code segment?

(A) 6

(B) 10

(C) 12

(D) 15

(E) Nothing is printed due to an infinite loop.

34. Consider the following class declarations.

```
public class Point
{
  private double x;  // x-coordinate
  private double y;  // y-coordinate

  public Point()
  {
    x = 0;
    y = 0;
  }

  public Point(double a, double b)
  {
    x = a;
    y = b;
  }

  // There may be instance variables, constructors, and methods that are not shown.
}


public class Circle
{
  private Point center;
  private double radius;

  /** Constructs a circle where (a, b) is the center and r is the radius.
   */
  public Circle(double a, double b, double r)
  {
    /* missing code */
  }
}
```

Which of the following replacements for `/* missing code */` will correctly implement the `Circle` constructor?

I.
```
center = new Point();
radius = r;
```

II.
```
center = new Point(a, b);
radius = r;
```

III.
```
center = new Point();
center.x = a;
center.y = b;
radius = r;
```

(A) I only

(B) II only

(C) III only

(D) II and III only

(E) I, II, and III

GO ON TO THE NEXT PAGE.

35. Consider the following code segment.

```
int num = 2574;
int result = 0;

while (num > 0)
{
  result = result * 10 + num % 10;
  num /= 10;
}
System.out.println(result);
```

What is printed as a result of executing the code segment?

(A) 2

(B) 4

(C) 18

(D) 2574

(E) 4752

36. Consider the following method.

```
public void test(int x)
{
  int y;

  if (x % 2 == 0)
    y = 3;
  else if (x > 9)
    y = 5;
  else
    y = 1;

  System.out.println("y = " + y);
}
```

Which of the following test data sets would test each possible output for the method?

(A) 8, 9, 12

(B) 7, 9, 11

(C) 8, 9, 11

(D) 8, 11, 13

(E) 7, 9, 10

GO ON TO THE NEXT PAGE.

37. Consider the following code segment.

```
int x = 1;
while ( /* missing code */ )
{
  System.out.print(x + " ");
  x = x + 2;
}
```

Consider the following possible replacements for `/* missing code */`.

I. `x < 6`

II. `x != 6`

III. `x < 7`

Which of the proposed replacements for `/* missing code */` will cause the code segment to print only the values `1 3 5` ?

(A) I only

(B) II only

(C) I and II only

(D) I and III only

(E) I, II, and III

GO ON TO THE NEXT PAGE.

38. Assume that `x` and `y` have been declared and initialized with `int` values. Consider the following Java expression.

```
(y > 10000) || (x > 1000 && x < 1500)
```

Which of the following is equivalent to the expression given above?

(A) `(y > 10000 || x > 1000) && (y > 10000 || x < 1500)`

(B) `(y > 10000 || x > 1000) || (y > 10000 || x < 1500)`

(C) `(y > 10000) && (x > 1000 || x < 1500)`

(D) `(y > 10000 && x > 1000) || (y > 10000 && x < 1500)`

(E) `(y > 10000 && x > 1000) && (y > 10000 && x < 1500)`

GO ON TO THE NEXT PAGE.

39. Consider the following recursive method.

```
public int recur(int n)
{
  if (n <= 10)
    return n * 2;
  else
    return recur(recur(n / 3));
}
```

What value is returned as a result of the call `recur(27)` ?

(A) 8

(B) 9

(C) 12

(D) 16

(E) 18

40. Consider the following recursive method.

```
public static void whatsItDo(String str)
{
  int len = str.length();
  if (len > 1)
  {
    String temp = str.substring(0, len - 1);
    whatsItDo(temp);
    System.out.println(temp);
  }
}
```

What is printed as a result of the call `whatsItDo("WATCH")` ?

(A)
```
WATC
WAT
WA
W
```

(B)
```
WATCH
WATC
WAT
WA
```

(C)
```
W
WA
WAT
WATC
```

(D)
```
W
WA
WAT
WATC
WATCH
```

(E)
```
WATCH
WATC
WAT
WA
W
WA
WAT
WATC
WATCH
```

END OF SECTION I

IF YOU FINISH BEFORE TIME IS CALLED,
YOU MAY CHECK YOUR WORK ON THIS SECTION.

DO NOT GO ON TO SECTION II UNTIL YOU ARE TOLD TO DO SO.
