Question 1

What is the sum of 1010110_2 and 110110_2

A. 10001000_2

B. 10001001_2

C. 10001010_2

D. 10011000_2

E. None of these

Question 2

Which of these puts into d the result of dividing a by b using floating point division?

A. `d = a / b;`

B. `d = (double)(a / b);`

C. `d = ((double)a) / b;`

D. Both B and C

E. A, B, and C

```
int a, b;
// code to initialize a and b

double d;
```

Question 3

What is output by the code to the right?

A. `30Hello30`

B. `1020Hello30`

C. `1020Hello2010`

D. `30Hello2010`

E. None of these

```
String s = "Hello";

int i = 10, j = 20;

System.out.print(i+j+s+j+i);
```

Question 4

Which of these replaces **<*1>** in the code to the right to cause "Invalid" to print when none of the other cases are met?

A. `otherwise:`

B. `case:`

C. `default:`

D. Nothing is needed

E. None of these

Question 5

Assume **<*1>** is filled in correctly. What is output if ch has the value `'c'`?

A. `3`

B. `34`

C. `34Invalid`

D. Nothing

E. None of these

```
switch(ch) {
case 'A':
case 'a':
  System.out.print("1"); break;
case 'B':
case 'b':
  System.out.print("2"); break;
case 'C':
case 'c':
  System.out.print("3"); break;
case 'D':
case 'd':
  System.out.print("4"); break;
<*1>
  System.out.print("Invalid");
}
```

Question 6

Which of these is not automatically initialized to `0` or `null` as appropriate?

A. instance variable

B. array element

C. local variable

D. All of these

E. None of these

Question 7

Suppose `Contact` has a public method called `getName()` which returns the name of a contact. Which of these returns the name of a properly declared and initialized `Contact c`?

A. `c.name.getName`

B. `c[0]`

C. `c.getName()`

D. `getName(c)`

E. None of these

Question 8

Which of these creates an `ArrayList` to which contacts can be added?

A. `ArrayList book();`

B. `ArrayList book = new ArrayList();`

C. `ArrayList<Contact> book;`

D.
```
ArrayList book =
                new ArrayList(Contact);
```

E. None of these

Question 9

Suppose `book` is declared correctly to be an `ArrayList` which holds objects of type `Contact`. Which of these sets `Contact c` to be the item at index 3 in `book`?

A. `c = book.get(3)`

B. `c = (Contact)(book.get(3))`

C. `c = ((Contact)book).get(3)`

D. Both A and C

E. None of these

```
// A class to represent a contact.  Assume
// the classes Name, PhoneNumber, and
// Address exist.

public class Contact {

  // constructors and methods not shown

  private Name name;
  private PhoneNumber home;
  private PhoneNumber office;
  private PhoneNumber cell;
  private PhoneNumber fax;
  private Address address;
}
```

Question 10

What is output by the code to the right?

A. `01234`

B. `010203040`

C. `12345`

D. `1020304050`

E. None of these

```
int i = 10;

int[] a = new int[5];

for (int j=0; j<5; ++j) {
  a[j] = i*j;
  System.out.print(a[j]);
}
```

Question 11

Which of these replaces **<*1>** in the code to the right to set i to a negative number if the key parameter is smaller than the data member key, 0 if they are the same, and a positive number if the parameter is bigger?

A. key.compareTo(this.key)

B. this.key.compareTo(key)

C. key.compareTo(key)

D. this.key.compareTo(this.key)

E. None of these

For the remaining questions assume **<*1>** is filled in correctly.

Question 12

Which of these types could be used as the type of the keys in a SimpleMap?

A. ArrayList

B. ListIterator

C. Double

D. int

E. Both C and D

Question 13

Suppose you wanted to add a method called elements() to the SimpleMap class that returned a two dimensional array of all of the keys and values in the map. In other words, each element of the outer array was a two element array containing a key and a value. What would be the proper prototype for this method?

A. void elements(Array[Object][2]);

B. Object[2][] elements();

C. Object[][2] elements();

D. Object[][] elements();

E. None of these

Question 14

Suppose n key/value pairs are added to a SimpleMap object in random order. What is the expected running time of a call to lookUp()? Choose the smallest correct answer.

A. O(1)

B. O(log n)

C. O(n)

D. O(n^2)

E. None of these

```
// A simple map class.  Objects can be
// added, but not removed.  Items are
// stored using a binary search tree.

public class SimpleMap {

  public SimpleMap(Comparable key,
                   Object value) {
    this.key = key;
    this.value = value;
  }

  public void add(Comparable key,
                  Object value) {
    int i = <*1>;
    if (i<0) {
      if (left == null)
        left = new SimpleMap(key,value);
      else left.add(key,value);
      }
    else if (i>0) {
      if (right == null)
        right = new SimpleMap(key,value);
      else right.add(key,value);
      }
  }

  public Object lookUp(Comparable key) {
    int i = <*1>;
    if (i==0) return value;
    else if (i<0) {
      if (left == null) return null;
      else return left.lookUp(key);
      }
    else {
      if (right == null) return null;
      else return right.lookUp(key);
    }
  }

  private Comparable key;
  private Object value;
  private SimpleMap left, right;
}
```

QUESTION 15

What is output by the code to the right?

A. `0`　　B. `1`

C. `15`　　D. Nothing

E. None of these

```
String s = "12345";

System.out.print(Integer.parseInt(s));
```

QUESTION 16

What is returned by `mangle("x")`?

A. `null`　　B. `""`

C. `"xxx"`　　D. `"x"`

E. None of these

QUESTION 17

What is returned by `mangle("0123456789")`?

A. `"0123456789"`　　B. `"069"`

C. `null`　　D. `"05789"`

E. None of these

```
public static String mangle(String s) {
  if ((s == null) || (s.equals("")))
    return "";
  else
    return s.charAt(0) +
               mangle(s.substring(
                        s.length()/2 + 1));
}
```

QUESTION 18

What is returned by `howMany(37)`?

A. `1`　　B. `2`　　C. `3`

D. `3`　　E. None of these

QUESTION 19

If `howMany(n)` returns `5` for a given `int n`, what is returned by `howMany(2*n)`?

A. `5`　　B. `10`　　C. `25`

D. `0`　　E. None of these

```
public static int howMany(int x) {
  if (x < 0) x = -x;
  int total = 0;
  while (x != 0) {
    if (x%2 == 1) total++;
    x /= 2;
  }
  return total;
}
```

QUESTION 20

When you evaluate boolean expressions joined with the `||` operator, in which case is only one side of the operator evaluated?

A. right side is true　　B. left side is true　　C. right side is false　　D. left side is false　　E. None of these

QUESTION 21

A value of which of these types may be stored in an `int` variable without using a cast?

A. `String`　　B. `double`　　C. `long`　　D. `Character`　　E. None of these

QUESTION 22

What is the value of `i` after executing the code to the right?

A. `2` B. `3` C. `4`

D. `5` E. None of these

```
int i;
int[] array = {10, 17, 24, 23, 30, 100};

for (i=0; i<array.length-1; ++i)
  if (array[i]>array[i+1]) break;
```

QUESTION 23

Which of the following replaces **<*1>** in the code to the right to check whether the character at position `i` in string `s` is a digit?

A. `s.charAt(i).isDigit()`

B. `Character.isDigit(s.charAt(i))`

C. `s.isDigit(i)`

D. `s.charAt.isDigit.i`

E. More than one of these

QUESTION 24

Assume **<*1>** is filled in correctly. What is returned by `process("ABC123")`?

A. `123` B. `6` C. `1`

D. `150` E. None of these

```
public static int process(String s) {
  int total = 0;
  for (int i=0; i<s.length(); ++i) {
    if (<*1>)
      total += s.charAt(i)-'0';
  }
  return total;
}
```

QUESTION 25

What replaces **<*1>** in the code to the right to get the first character of an item dequeued from `q`?

A. `q.dequeue().charAt(0)`

B. `q.dequeue()[0]`

C. `((String)q.dequeue())[0]`

D. `((String)q.dequeue()).charAt(0)`

E. More than one of these

QUESTION 26

Assume **<*1>** is filled in correctly. What is output by the code to the right?

A. `MSH` B. `HSM` C. `HHH`

D. `MMM` E. None of these

```
// Assume a Queue class has been written
// which enqueues Objects with enqueue()
// and dequeues Objects with dequeue(),
// returning the object dequeued

Queue q = new Queue();

q.enqueue("Happy");

q.enqueue("Sad");

q.enqueue("Mad");

for (int i=0; i<3; ++i)
  System.out.print(<*1>);
```

QUESTION 27

What replaces **`<*1>`** in the code to the right to specify that `PENNY`, `NICKEL`, and `DIME` are publicly accessible class constants?

A. `public final`

B. `protected static final`

C. `public static final`

D. `final static protected`

E. None of these

For the remaining questions assume **`<*1>`** is filled in correctly.

QUESTION 28

Which of these builds a `MoneyGame` where the players start with 4 pennies, 3 nickels, and 2 dimes?

A. `new MoneyGame(2,3,4)`

B. `new MoneyGame(4,3,2)`

C. `new MoneyGame(int[]{2,3,4})`

D. `new MoneyGame(int[]{4,3,2})`

E. More than one of these

QUESTION 29

Suppose `MoneyGame mg` is a reference to the `MoneyGame` in the previous problem. What is returned by `mg.getTurn()` if it is called before any calls to `mg.makeMove()`?

A. `0` B. `1` C. `true`

D. `false` E. None of these

QUESTION 30

Suppose `mg` is a correctly initialized `MoneyGame`. Which of these method calls represents the current player putting a nickel in the pot and taking three pennies from the pot?

A. `mg.makeMove(MoneyGame.NICKEL, 3, MoneyGame.PENNY)`

B. `mg.makeMove(5,3,0)`

C. `mg.makeMove(MoneyGame.NICKEL, 3, 0)`

D. `mg.makeMove(--MoneyGame.NICKEL, MoneyGame.PENNY+=3)`

E. None of these

```
public class MoneyGame {

  public MoneyGame(int p, int n, int d) {
    players = new int[2][3];
    pot = new int[3];
    players[0][PENNY] =
       players[1][PENNY] = p;
    players[0][NICKEL] =
       players[1][NICKEL] = n;
    players[0][DIME] =
       players[1][DIME] = d;
    turn = 0;
  }

  public boolean done() {
    return (players[turn][PENNY] == 0 &&
            players[turn][NICKEL] == 0 &&
            players[turn][DIME] == 0);
  }

  public void makeMove(int coin, int p,
                                   int n) {
    if (players[turn][coin] == 0 ||
        pot[PENNY]<p || pot[NICKEL]<n ||
        (coin == PENNY && (p + 5*n)>0) ||
        (coin == NICKEL && (p + 5*n)>4) ||
        (coin == DIME && (p + 5*n)>9))
      throw new IllegalArgumentException();
    else {
      pot[coin]++;
      pot[PENNY]-=p;
      pot[NICKEL]-=n;
      players[turn][coin]--;
      players[turn][PENNY]+=p;
      players[turn][NICKEL]+=n;
      turn = 1-turn;
    }
  }

  public int getTurn() { return turn; }

  <*1> int PENNY = 0;
  <*1> int NICKEL = 1;
  <*1> int DIME = 2;

  private int[][] players;
  private int[] pot;
  private int turn;
}
```

Question 31

What replaces **`<*1>`** in the code to the right to return the square of private data member `side`?

A. `return side ^ 2;`

B. `return side ** 2;`

C. `return side * side;`

D. Both A and C

E. None of these

Question 32

What replaces **`<*2>`** in the code to the right to compute the square root of two?

A. `root(2,2)` B. `Math.root(2,2)`

C. `sqrt(2)` D. `Math.sqrt(2)`

E. More than one of these

```
public class Square {

  public Square(int side) {
    if (side>0) this.side = side;
  }

  public int area() { <*1> }

  public int perimeter() { return 4*side; }

  public double diagonal() {
    return <*2> * side;
  }

  private int side;
}
```

Question 33

Which of the following loops will output exactly four stars?

I
```
for (int z=4;z>1;--z)
  System.out.print('*');
```

II
```
int z=1;
do {
  System.out.print('*');
} while (z<4);
```

III
```
int z=10000;
while ((z/=10)!=0)
 System.out.print('*');
```

A. I only B. II only C. III only D. I, II, and III E. None of these

Question 34

What replaces **`<*1>`** in the code to the right to check whether the item in the array at index `i` has the same reference as the `item` parameter?

A. `array[i] == item`

B. `array[i] = item`

C. `array[i] equals item`

D. `array[i].equals(item)`

E. None of these

Question 35

Assume **`<*1>`** has been filled in correctly. What algorithm is implemented by `mystery()`?

A. Binary search B. Merge sort

C. Multiplication D. Inverse

E. None of these

```
public static int mystery(Object[] array,
                          Object item) {
  int i = 0;
  while (i < array.length) {
    if (<*1>)
      return i;
    ++i;
  }
  return -1;
}
```

Question 36

What replaces **<*1>** in the code to the right so that the loop prints all of the elements of the collection?

A. `c(iter)`

B. `c.iter.next()`

C. `Iterator.next(iter)`

D. `iter.next()`

E. None of these

```
public static void output(Collection c) {
  Iterator iter = c.iterator();

  try {
    while(true)
      System.out.print(<*1>);
  }

  catch(NoSuchElementException e) {}
}
```

Question 37

Assume **<*1>** is filled in correctly. What method is called inside the `print()` method on the objects in the collection to convert them to a type appropriate to display on the screen?

A. `equals()`   B. `toString()`

C. `clone()`   D. `hashCode()`

E. None of these

Question 38

What replaces **<*1>** in the code to the right to indicate that `class B` implements `interface A`?

A. `implements A`   B. `extends A`

C. `interface A`   D. `has A`

E. None of these

```
public interface A {
  public void someMethod();
}

public class B <*1> {
  public void someMethod() {
    System.out.print("Hello");
  }
}
```

Question 39

Assume **<*1>** is filled in correctly. Which of the following declarations is legal?

A. `B obj = new B();`

B. `A obj = new A();`

C. `B obj = new A();`

D. `A obj = new B();`

E. Both A and D

Question 40

What number is formed from the bitwise XOR of the binary representations of 24 and 37?

A. `72`   B. `61`   C. `39`   D. `10`   E. None of these