University Interscholastic League
Computer Science Competition

Number 149 (Invitational A - 2015)

General Directions:

1) **DO NOT OPEN EXAM UNTIL TOLD TO DO SO.**

2) **NO CALCULATOR OF ANY KIND MAY BE USED.**

3) There are 40 questions on this contest exam. You have 45 minutes to complete this contest. If you are in the process of actually writing an answer when the signal to stop is given, you may finish writing that answer.

4) Papers may not be turned in until 45 minutes have elapsed. If you finish the test before the end of the allotted time, remain at your seat and retain your paper until told to do otherwise. Use this time to check your answers.

5) All answers must be written on the answer sheet/Scantron card provided. Indicate your answers in the appropriate blanks provided on the answer sheet or on the Scantron card. Clean erasures are necessary for accurate Scantron grading.

6) You may place as many notations as you desire anywhere on the test paper, but not on the answer sheet or Scantron card, which are reserved for answers only.

7) You may use additional scratch paper provided by the contest director.

8) All questions have ONE and only ONE correct (BEST) answer. There is a penalty for all incorrect answers.

9) A reference to commonly used Java classes is provided at the end of the test, and you may use this reference sheet during the contest. You may detach the reference sheets from the test booklet, but DO NOT DO SO UNTIL THE CONTEST BEGINS.

Scoring:

1) All questions will receive 6 points if answered correctly; no points will be given or subtracted if unanswered; 2 points will be deducted for an incorrect answer.

Note: Correct responses are based on Java, **J2sdk v 1.7.25**, from Sun Microsystems, Inc. All provided code segments are intended to be syntactically correct, unless otherwise stated (i. e. `error` is an answer choice) and any necessary Java 2 Standard Packages have been imported. Ignore any typographical errors and assume any undefined variables are defined as used. **For all output statements, assume that the System class has been statically imported…** ***import static java.lang.System.*;***

QUESTION 1

Which of these is NOT equivalent to 88_{16} - 206_{8} ?

A. 2_{16} B. 2_{8} C. 2_{2} D. 2_{10} E. All are equivalent

QUESTION 2

What is the result of the expression to the right?

A. -4 B. -1.7

C. -2 D. 3 E. 3.6

```
23 / 10 % 3 - 4
```

QUESTION 3

What is output by the code to the right?

A.
```
----*----*
9.5
```
B.
```
----*----*
9.50
```
C.
```
----*----*
 9.5
```
D.
```
----*----*
 9.50
```
E.
```
----*----*
    9.50
```

```
System.out.println("----*----*");
System.out.printf("%5.2f",9.5);
```

QUESTION 4

What is output by the code to the right?

A. UILCOMPUTERSCONCE2015

B. UILCOMPUTERSCIENCE2015

C. UOLCOMPUTORSCONCO2015

D. UOLCOMPUTORSCOONCO2015

E. OUOOLOCOOOMOPOUOTOOROSOCOONOCOO2O0O1O5O

```
String s = "UILCOMPUTERSCIENCE2015";
s = s.replaceAll("[IE]+","O");
out.println(s);
```

QUESTION 5

What is output by the code to the right?

A. true B. false

```
boolean p = true;
boolean q = false;
out.println(!(p||q));
```

QUESTION 6

What is output by the code to the right?

A. 50 B. 10.0

C. 10 D. 10000000000

E. 50.0

```
int x = 100;
double y = 0.5;
out.println(Math.pow(x,y));
```

QUESTION 7

What is output by the code to the right?

A. `34` B. `35.0` C. `34.4` D. `35`

E. There is no output due to an error.

```
long g = 42;
double d = 7.6;
g -= d;
out.printf("%d",g);
```

QUESTION 8

What is output by the code to the right?

A. `-4` B. `-1` C. `4` D. `1`

E. There is no output due to an error.

```
int a = 25;
if(a%-6>0)
 out.println(a%-6);
else
 out.println(a/-6);
```

QUESTION 9

What is output by the code to the right?

A. `11` B. `5.2` C. `9` D. `8`

E. There is no output due to an error.

```
int x = 100,y = 0;
while((x/=3)>0)
 y+=2;
out.println(x+y);
```

QUESTION 10

What is output by the code to the right?

A. `8` B. `2` C. `3` D. `7`

E. There is no output due to an error.

```
int list[]={-5,-6,4,2,-3,7};
out.println(list[list[4]]
           +list.length);
```

QUESTION 11

Below is a value in a data file called **`"stuff.dat"`**.

`5.2`

In the code segment to the right, which choice is best for **<statement 1>** in order to retrieve the data for calculation purposes?

A. `double d = f.nextInt();`

B. `double d = f.nextDouble();`

C. `int n = f.nextDouble();`

D. `int n = f.nextInt();`

E. All statements will work properly

```
Scanner f = new Scanner(new
File("stuff.dat"));
<statement 1>
```

QUESTION 12

What is output by the code to the right?

A. `62 64` B. `62 32`

C. `126 64` D. `126 128`

E. There is no output due to an error.

```
int a = 0, b = 1;
do{
 b*=2;
 a+=b;
}
while(b<50);
out.println(a+" "+b);
```

QUESTION 13

To the right are three lines taken from the Java Order of Precedence chart. Which choice represents the correct order of precedence for these three lines?

A. III, I, II B. I, II, III C. I, III, II

D. III, II, I E. II, I, III

I. `&`
II. `&&`
III. `< > <= >= instanceof`

QUESTION 14

Which of the choices listed to the right represents the correct order from greatest to least of the bit storage capacity for the data types listed?

A. I, III, IV, II, V B. V, IV, III, II, I

C. V, II, III, IV, I D. I, II III, IV, V

E. None of these

I. char
II. double
III. float
IV. int
V. long

QUESTION 15

What is output by the code to the right?

A. 0 1 2 2 4 6 8

B. 0 1 2 0 1 2

C. 0 1 2 2 4 6

D. There is no output due to a compile error.

E. There is no output due to a runtime error.

```
Integer [] list={0,1,2};
ArrayList<Integer> aList = new
ArrayList<Integer>();
aList.add(2);aList.add(4);
aList.add(6);aList.add(8);
for(Integer x:list)
 out.print(x+" ");
list = aList.toArray(list);
for(Integer x:list)
 out.print(x+" ");
```

QUESTION 16

Using the mergeSort code to the right, what is output by the client code below?

```
int[]  list = {5,7,3,9,4,6};
mergeSort(list);
outputList(list);
```

A. 9 7 6 5 4 3

B. 3 4 5 6 7 9

C. 5 7 3 9 4 6

D. 6 4 9 3 7 5

E. not possible to determine

QUESTION 17

In the code to the right, which of the lines below the five indicated <statements> needs to be altered in order to sort a list in descending order?

A. <statement 1>
B. <statement 2>
C. <statement 3>
D. <statement 4>
E. <statement 5>

QUESTION 18

There are nine <doc> comments in the code to the right, explaining the purpose of the code just below each one. Which of the choices below is NOT correct relating to these <doc> statements?

A. <doc 4> merge sort call for last half of list
B. <doc 1> initial merge sort call
C. <doc 2> find the middle position of the current list
D. <doc 7> clean up remaining second half elements, if any
E. <doc 9> transfer elements from temporary list to original list

QUESTION 19

What is the least restrictive running time for the worst case scenario for merge sort algorithm?

A. O(1) B. O(log N) C. O(N^2)

D. O(N) E. O(N log N)

```
public static void mergeSort(int[] list){
 int n = list.length;
 int[] temp = new int[n];
 //<doc 1>
 mergeSortHelper(list, 0, n - 1, temp);
}
private static void mergeSortHelper(int[] list,
 int front, int back, int[] temp)
{
 //<statement 1>
 if (front < back){
 //<doc 2><statement 2>
 int mid = (front + back) / 2;
 //<doc 3>
 mergeSortHelper(list, front, mid, temp);
 //<doc 4>
 mergeSortHelper(list, mid + 1, back, temp);
 //<doc 5>
 merge(list, front, mid, back, temp);
 }
}
private static void merge(int[] list,
 int front, int mid, int back, int[] temp){
  int i = front;
  int j = mid + 1;
  int k = front;
  //<doc 6><statement 3>
  while (i <= mid && j <= back){
    <statement 4>
    if (list[i] < list[j]){
        temp[k] = list[i];
        i++;
    }
    else{
       temp[k] = list[j];
       j++;
    }
    k++;
  }
  //<doc 7><statement 5>
  while (i <= mid){
     temp[k] = list[i];
     k++;
     i++;
  }
 //<doc 8>
 while (j <= back) {
    temp[k] = list[j];
    j++;
    k++;
  }
 //<doc 9>
 for(int x=front;x<=back;x++)
    list[x]=temp[x];
}
public static void outputList(int[]list){
 for(int x=0;x<list.length;x++)
    out.print(list[x]+" ");
 out.println();
}
```

QUESTION 20

The two's complement system is all about representing negative numbers in binary. For example, the positive value 72 in 8-bit binary is **01001000**. To find the binary representation for -72 using two's complement, you use this easy conversion process. Start from the right and keep all zeroes the same until you reach the first 1 digit. Keep that 1 the same also, and flip everything else, with an 8-bit binary result of **10111000** for -72.With that in mind, which of the following choices represents the decimal equivalent of the two's complement binary value **10010101**?

A. -109 B. -106 C. -107 D. -105 E. -108

QUESTION 21

What is output by the code to the right?

A. `11` B. `59` C. `59.5` D. `11.5`

E. There is no output due to an error.

```
int x = 31;
char a = 48;
double d = 19.5;
out.println(x-d+a);
```

QUESTION 22

What is output by the client code to the right?

A.
```
Power = electric
Power = electric
Power = electric
```

B.
```
Power = motor
Power = motor
Power = electric
```

C.
```
Power = motor
Power = electric
Power = electric
```

D.
```
Power = motor
Power = motor
Power = motor
```

E. None of these

QUESTION 23

Which choice best describes statement 1 in the client code to the right?

A. downcasting B. supercasting

C. early binding D. subcasting

E. late binding

QUESTION 24

Which choice best describes statement 2 in the client code to the right?

A. downcasting B. supercasting

C. early binding D. subcasting

E. late binding

QUESTION 25

Which choice best describes statement 3 in the client code to the right?

A. downcasting B. supercasting

C. early binding D. subcasting

E. late binding

```
class Vehicle
{
 public String power = "motor";
 public void view()
 {
  out.println("Power = "
                  + power);
 }
}
class Car extends Vehicle
{
 public String power = "electric";
 public Car(String power)
 {
  this.power = power;
 }
 public void view()
 {
  out.println("Power = "
                   + power);
 }
}
Vehicle car = new Car("electric");
//statement 1
out.println("Power = " + car.power);
//statement 2
out.println("Power = " +
((Car)car).power);
//statement 3
car.view();
```

QUESTION 26

What is output by the code to the right?

A. `HAPPY*N*Y*A*!*` B. `HAPPY N* Y*A*!`

C. `HAPPY*N***Y*A*!*` D. `HAPPY*N*Y*A*!`

E. There is no output due to an error

```
String s = "HAPPY NEW YEAR!";
args=s.split("[B-E R-W]+");
for(String t:args)
 out.print(t+"*");
```

QUESTION 27

What is output by the code to the right?

A. `012340240304` B. `02134024030405`

C. `012340123401234012340123401234`

D. `0123402403042` E. There is no output due to an error.

```
char[]list={'0','1','2','3','4'};
for(int x=1;x<=list.length;x++)
 for(char a:list)
    if(a%x==0)
       out.print(a);
```

QUESTION 28

What is output by the client code below?

```
//client code
ListNode ln = new ListNode();
ln = new ListNode(3,ln);
ln = new ListNode(5,ln);
ListNode m = ln;
while(m!=null)
{
 out.print(m.val+" ");
 m=m.next;
}
```

A. `0 0 0` B. `5 3 0`

C. `null null null` D. `5 3`

E. `0 3 5`

```
class ListNode
{
 public ListNode()
 {
    val = 0;
    next = null;
 }
 public ListNode(int v, ListNode n)
 {
    val = v;
    next = n;
 }
 public int val;
 public ListNode next;
}
```

QUESTION 29

What is output by the code to the right?

A. `3 4 4.0` B. `4 3 4.0`

C. `4 3 5.6` D. `3 4 5.6`

E. There is no output due to an error.

```
double [][] dubs = {{7.3,4.5,2.7},
                    {3.4,5.6,7.8},
                    {1.2,7.3,4.0},
                    {5.2,3.6,4.9}};
out.println(dubs.length+" "+
    dubs[3].length +" "+dubs[2][2]);
```

QUESTION 30

What is output by the client code 1 to the right?

A. `3 1 5` B. `1 2 4`

C. `1 5 3` D. `0 5 3`

E. There is no output due to an error.

QUESTION 31

What is output by the client code 2 to the right?

A. `3 5 -5` B. `-3 5 -5`

C. `4 2 8` D. `-3 5 5`

E. There is no output due to an error.

```
public static int mystNum(int x, int y, int z)
{
 x=x/y+z;
 y=x/y+z;
 z=x/y+z;
 return x/y+z;
}

//client code 1
int x=8,y=2,z=-1;
out.print(mystNum(x,y,z)+" ");
x=10;y=5;z=3;
out.print(mystNum(x,y,z)+" ");
x=10;y=1;z=3;
out.println(mystNum(x,y,z));

//client code 2
int x=5,y=2,z=-1;
out.print(mystNum(x,y,z)+" ");
x=8;y=5;z=3;
out.print(mystNum(x,y,z)+" ");
x=9;y=3;z=-1;
out.println(mystNum(x,y,z));
```

QUESTION 32

Using the generic stack pseudocode to the right, what was the last value popped, and which item is left at the top of the stack ?

A. `7 3` B. `3 7`

C. `7 4` D. `4 7`

E. None of these

```
Push 9
Push 3
Push 5
Pop x
Push 4
Pop x
Pop x
Push 7
```

QUESTION 33

How many ordered triples make this boolean expression true? $\overline{A+B}+C$

A. 5 B. 3 C. 6 D. 4 E. 2

QUESTION 34

Infix notation is the kind normally used in algebraic expressions, such as **`3 + 5 * 6`**, where the operators are between the operands. However, there is also prefix notation, where the operators are before the operands, such as **`+ 3 * 5 6`**, and postfix notation, operators after operands, like this: **`3 5 6 * +`**. Notice carefully that the operands never move around: only the operators change places.

Here is another example: the infix expression **`6 * 7 + 9 - 8 * 2`** translates the prefix expression **`- + * 6 7 9 * 8 2`**, and **`6 7 * 9 + 8 2 * -`** for postfix.

Given these examples to examine and study carefully, which of the infix expressions below matches the postfix expression shown?

`3 9 * 6 - 5 2 ^ +`

A. **`3 - 9 * 6 + 5 ^ 2`** B. **`3 * 9 + 6 - 5 ^ 2`** C. **`3 * 9 - 6 + 5 ^ 2`**

D. **`9 * 6 - 5 ^ 2 + 3`** E. None of these

QUESTION 35

What is output by the code to the right?

A. `0` B. `00000014`

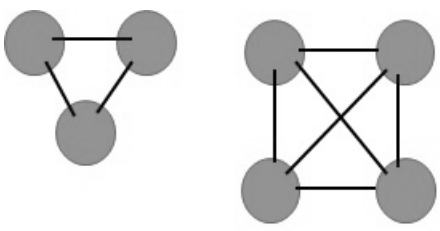
C. `00000000` D. `14.00000`

E. `14.0`

```
byte c = 100;
c>>=7;
out.println(Integer.toBinaryString(c));
```

QUESTION 36

Below are two complete graphs. The three node graph has three edges, and the four node graph has 6 edges. How many edges would a six-node complete graph have?



A. 8 B. 12 C. 15 D. 9 E. None of these

QUESTION 37

What is output by the code to the right?

A. `[1.2, 3.4, 5.6, 9.4][1.1, 1.2, 3.4, 5.6, 9.4]`
B. `[1.2, 5.6, 3.4, 9.4][1.1, 1.2, 3.4, 9.4, 5.6]`
C. `[1.2, 3.4, 5.6, 9.4][1.2, 3.4, 5.6, 9.4, 1.1]`
D. `[3.4, 5.6, 1.2, 9.4][3.4, 5.6, 1.2, 9.4, 1.1]`
E. None of these

```
double[]list = {3.4, 5.6, 1.2, 9.4};
ArrayList<Double>dList= new
 ArrayList<Double>();
for(double d:list)
 dList.add(d);
PriorityQueue<Double> pqd = new
 PriorityQueue<Double>(dList);
out.print(pqd);
pqd.add(1.1);
out.println(pqd);
```

QUESTION 38

Which of the following logical statements is represented by the digital electronics diagram shown?

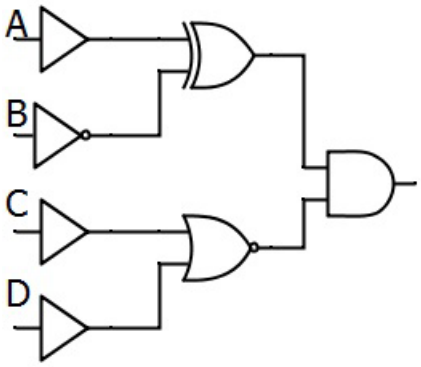
A. $A \oplus \overline{B} * \overline{C+D}$

B. $(A+\overline{B}) * \overline{C \oplus D}$

C. $A \oplus \overline{B} + \overline{C * D}$

D. $A * \overline{B} + \overline{C \oplus D}$

E. None of these



QUESTION 39

Find f(20) according to the recursive function definition shown below.

f(20) =

```
f(x)= f(x-4)+1     when x>5
f(x)= 2            otherwise
```

QUESTION 40

Simplify this expression to have only two operators and one NOT. The allowable operators include AND(*), OR(+), XOR(⊕), and NOT (over bar).

$$\overline{(A+\overline{B * \overline{C}}) * \overline{(A * (B * C))}}$$

Standard Classes and Interfaces — Supplemental Reference

class java.lang.Object
- `boolean equals(Object other)`
- `String toString()`
- `int hashCode()`

interface java.lang.Comparable<T>
- `int compareTo(T other)`
  Return value < 0 if `this` is less than `other`.
  Return value = 0 if `this` is equal to `other`.
  Return value > 0 if `this` is greater than `other`.

class java.lang.Integer implements Comparable<Integer>
- `Integer(int value)`
- `int intValue()`
- `boolean equals(Object obj)`
- `String toString()`
- `int compareTo(Integer anotherInteger)`
- `static int parseInt(String s)`
- `static int parseInt(String s, int radix)`

class java.lang.Double implements Comparable<Double>
- `Double(double value)`
- `double doubleValue()`
- `boolean equals(Object obj)`
- `String toString()`
- `int compareTo(Double anotherDouble)`
- `static double parseDouble(String s)`

class java.lang.String implements Comparable<String>
- `int compareTo(String anotherString)`
- `boolean equals(Object obj)`
- `int length()`
- `String substring(int begin, int end)`
  Returns the substring starting at index `begin` and ending at index `(end - 1)`.
- `String substring(int begin)`
  Returns `substring(from, length())`.
- `int indexOf(String str)`
  Returns the index within this string of the first occurrence of `str`. Returns `-1` if `str` is not found.
- `int indexOf(String str, int fromIndex)`
  Returns the index within this string of the first occurrence of `str`, starting the search at the specified index.. Returns `-1` if `str` is not found.
- `charAt(int index)`
- `int indexOf(int ch)`
- `int indexOf(int ch, int fromIndex)`
- `String toLowerCase()`
- `String toUpperCase()`
- `String[] split(String regex)`
- `boolean matches(String regex)`

class java.lang.Character
- `static boolean isDigit(char ch)`
- `static boolean isLetter(char ch)`
- `static boolean isLetterOrDigit(char ch)`
- `static boolean isLowerCase(char ch)`
- `static boolean isUpperCase(char ch)`
- `static char toUpperCase(char ch)`
- `static char toLowerCase(char ch)`

class java.lang.Math
- `static int abs(int a)`
- `static double abs(double a)`
- `static double pow(double base, double exponent)`
- `static double sqrt(double a)`
- `static double ceil(double a)`
- `static double floor(double a)`
- `static double min(double a, double b)`
- `static double max(double a, double b)`
- `static int min(int a, in b)`
- `static int max(int a, int b)`
- `static long round(double a)`
- `static double random()`
  Returns a `double` value with a positive sign, greater than or equal to `0.0` and less than `1.0`.

interface java.util.List<E>
- `boolean add(E e)`
- `int size()`
- `Iterator<E> iterator()`
- `ListIterator<E> listIterator()`
- `E get(int index)`
- `E set(int index, E e)`
  Replaces the element at `index` with the object `e`.
- `void add(int index, E e)`
  Inserts the object `e` at position `index`, sliding elements at position `index` and higher to the right (adds 1 to their indices) and adjusts `size`.
- `E remove(int index)`
  Removes element from position `index`, sliding elements at position `(index + 1)` and higher to the left (subtracts 1 from their indices) and adjusts `size`.

class java.util.ArrayList<E> implements List<E>

class java.util.LinkedList<E> implements List<E>, Queue<E>

Methods in addition to the `List` methods:
- `void addFirst(E e)`
- `void addLast(E e)`
- `E getFirst()`
- `E getLast()`
- `E removeFirst()`
- `E removeLast()`

class java.util.Stack<E>
- boolean isEmpty()
- E peek()
- E pop()
- E push(E item)

interface java.util.Queue<E>
- boolean add(E e)
- boolean isEmpty()
- E peek()
- E remove()

class java.util.PriorityQueue<E>
- boolean add(E e)
- boolean isEmpty()
- E peek()
- E remove()

interface java.util.Set<E>
- boolean add(E e)
- boolean contains(Object obj)
- boolean remove(Object obj)
- int size()
- Iterator<E> iterator()
- boolean addAll(Collection<? extends E> c)
- boolean removeAll(Collection<?> c)
- boolean retainAll(Collection<?> c)

class java.util.HashSet<E> implements Set<E>

class java.util.TreeSet<E> implements Set<E>

interface java.util.Map<K,V>
- Object put(K key, V value)
- V get(Object key)
- boolean containsKey(Object key)
- int size()
- Set<K> keySet()
- Set<Map.Entry<K, V>> entrySet()

class java.util.HashMap<K,V> implements Map<K,V>

class java.util.TreeMap<K,V> implements Map<K,V>

interface java.util.Map.Entry<K,V>
- K getKey()
- V getValue()
- V setValue(V value)

interface java.util.Iterator<E>
- boolean hasNext()
- E next()
- void remove()

interface java.util.ListIterator<E> extends java.util.Iterator<E>

Methods in addition to the `Iterator` methods:
- void add(E e)
- void set(E e)

class java.lang.Exception
- Exception()
- Exception(String message)

class java.util.Scanner
- Scanner(InputStream source)
- boolean hasNext()
- boolean hasNextInt()
- boolean hasNextDouble()
- String next()
- int nextInt()
- double nextDouble()
- String nextLine()
- Scanner useDelimiter(String pattern)

Computer Science Answer Key
UIL Invitational A 2015

1) C	11) B	21) C	31) B
2) C	12) C	22) C	32) B
3) D	13) A	23) C	33) A
4) C	14) C	24) A	34) C
5) B	15) A	25) E	35) A
6) B	16) B	26) A	36) C
7) A	17) D	27) D	37) B
8) D	18) D	28) B	38) A
9) D	19) E	29) B	39) 6
10) E	20) C	30) E	40) $B(\overline{A \oplus C})$

Note: Since AND and XOR have the commutative property, any answer that is a correctly commuted version of this answer is correct.

Note to Graders:

- All provided code segments are intended to be syntactically correct, unless otherwise stated (e.g. error is an answer). **Ignore any typographical errors.**
- Any necessary Standard Java 2 Packages are assumed to have been imported as needed.
- Assume any undefined (undeclared) variables have been defined as used.

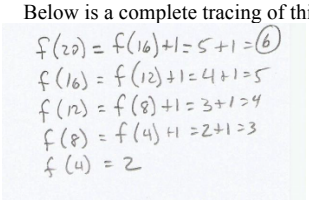
Explanations:

1. C $88_{16} - 206_8 = 136_{10} - 134_{10} = 2_{10} = 2_8 = 2_{16} = 10_2$
2. C 23 / 10 % 3 - 4 = 2 % 3 - 4 = 2 - 4 = -2
3. D The 5 designates the total field width, including the decimal places and the period, and the 2 shows the number decimal places to show. That leaves only two places for the whole number portion.
4. C The "[IE]+" pattern finds "E", "I", and any multiple sequence containing "E" or "I" and replaces it with a single "O".
5. B Since p is true and q is false, p OR q is true, which makes the NOT of P and Q false.
6. B The square root of 100 (100 to the power of 1/2) is 10.0
7. A 42 - 7.6 = 34.4, which truncates to 34 in the autocast provided by -=
8. D The value of 25 % -6 is 1, not -1 as you might think, therefore the output is 1.
9. D The trace values of x and y are 100 and 0, 33 and 2, 11 and 4, 3 and 6, and finally 0 and 8, with an output of 8.
10. E The value in position 4 is -3, which when used as an index value causes an ArrayIndexOutOfBoundsException runtime error since there is no position -3 in the array.
11. B Since nextDouble() retrieves a double value, which can only be assigned to a double variable, the only choice that will work is `double d = f.nextDouble();`
12. C The sequence for a and b is: 0 1, 2 2, 6 4, 14 8, 30 16, 62 32, 126 64
13. A The **< > <= >= instanceof** operators are on line 6 of the chart, followed by **&** on line 8, and **&&** on line 11.
14. C The long(V) and double(II) data types both use 64 bits, float(III) and int(IV) both 32, and char(I) 16 bits of storage.
15. A The toArray method creates new memory for the ***list*** array since the original capacity of ***list*** is too small, therefore this method works just fine.
16. B This is the merge sort, which sorts a list of numbers in ascending order. The output is simply the choice that shows all of the numbers in order from least to greatest.
17. D The comparison operator in `<statement 4>` needs to be reversed to be `<if (list[i] > list[j])>` in order to sort the list in descending order.
18. D `<doc 7>` should `be` **`clean up remaining first half elements, if any`**
19. E The running time for a merge sort in any case is O(N log N).
20. C To convert back from two's complement, use the same process as described to find the positive value, then just make it negative. 10010101 converts back to 01101011, which is the value 107, hence the original bit string is -107.
21. C 31 - 19.5 + 48 is equal to 59.5.
22. C
23. C
24. A
25. E With inheritance, early (static) binding occurs at compile time, while late (dynamic) binding occurs at run time. If at compile time the compiler can make a decision about what to use, it will, which is called **early binding**. The method call in statement 1 uses the power field of the super class (Vehicle), an example of this. Statement 2 is an example of **downcasting**, which forces the compiler to use the subclass instance field. If it is not possible for the compiler to decide due to ambiguity at compile time, **late binding** occurs, as in statement 3, where the view() method called is the one that belongs to the subclass (Car) class.
26. A The split pattern `"[B-E R-W]+"` means split on one or more (+) of any capital letters between B and E, R and W, or the space character.
27. D Since the values of '0' through '4' are 48, 49, 50, 51, and 52, the characters are output only when these values are divisible by the values 1 through 5. They are all divisible by 1, hence "01234". Only 48, 50, and 52 are divisible by 2, producing "024", and so on.
28. B This simple example of a linked list creates three nodes with instance field values of 0, 3, and 5, linking them together in reverse order. The while loop traverses the list and outputs each value.
29. B There are four rows, three columns in row 3, and the value in row 2, column 2 is 4.0 (zero indexing makes the first row and first column at position [0][0])
30. E A division by zero error occurs in the first method call when x, y, and z start at 8, 2, and -1, become 3, 2, -1 in the first assignment statement, and then 3, 0, -1 in the second statement, causing a run time error in the third assignment statement of the method.
31. B The sequences of values for each method call are:
```
5 2 -1, 1 2 -1, 1 -1 -1, 1 -1 -2 = -3
8 5 3, 4 5 3, 4 3 3, 4 3 4 = 5
9 3 -1, 2 3 -1, 2 -1 -1, 2 -1 -3 = -5
```
32. B The stack contents sequence is as follows, with the end of the list being the top of the stack: [9], [9, 3], [9, 3, 5], [9, 3], [9, 3, 4], [9, 3], [9], [9, 7]. The last value popped was 3, and 7 is at the top of the stack.
33. A

A	B	C	A+B	$\overline{A+B}$	$\overline{A+B}+C$
0	0	0	0	1	1
0	0	1	0	1	1
0	1	0	1	0	0
0	1	1	1	0	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	0	0
1	1	1	1	0	1

34. C The first rule is that the operands stay in the same order, 3 9 6 5 2. Then insert the operators as shown. The * is immediately after the 3 and 9, so it goes between them, likewise with the ^ between the 5 and 2. The - is after the 3*9 and 6, so it goes between them, and finally the + goes between the 3*9-6 and 5^2 to complete the expression.
35. A The value 100 right shifted 7 times equals zero. This is the equivalent of 100 /2/2/2/2/2/2/2, which in sequence equals 50, 25, 12, 6, 3, 1, and finally 0. The toBinaryString method only shows significant digits, therefore leading zeroes are not shown.

36. C The function of the pattern for the number of edges for any complete graph is N(N-1)/2, so for three nodes, 3*2/2 = 3, four nodes is 4*3/2=6, five nodes 5*4/2=10, and six nodes 6*5/2=15.
37. B In a Priority Queue, the elements are stored in a min heap, as is shown in the first output. When an element is added, it "finds" itself in the natural order of the list, again within the structure of the min heap. order for that data type.
38. A The signals A and NOT B go into a NXOR gate, which feeds into an AND gate, which also receives the NOT OR signal from C and D.
39. 6 Below is a complete tracing of this recursive function call shown above.



40. $\overline{B(A \oplus C)}$

Explanation: In the diagram below, DeMorgan's law is used to break the NOT over the OR and the AND, resulting in two sets of double NOTs "flying away". In the resulting expression, B is factored out, leaving the simplification of NXOR, which when returned to NXOR results in the final simplified expression, which has two gates, AND, XOR, and one NOT over the XOR.