University Interscholastic League
Computer Science Competition

Number 143 (Invitational A - 2014)

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 $11110_2 + 11011_2$?

A. 57_{10} B. 71_8 C. 39_{16} D. 111101_2 E. All are equivalent

QUESTION 2

What is output by the code to the right?

A. `19 32` B. `32 13`

C. `19 13` D. `32 19`

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

```
long b = 19;
int c = 13;
b+=c;
out.println(b+" "+c);
```

QUESTION 3

What is output by the code to the right?

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

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

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

```
Integer [] list = {1,2,3,4.0};
out.println(list[3]);
```

QUESTION 4

What is output by the code to the right?

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

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

E. There is no output.

```
int j = 5;
do
{
  out.print(--j + " ");
}
while (j>1);
```

QUESTION 5

What is output by the code to the right?

A. `gBad` B. `reak` C. `Brea` D. `a` E. `k`

```
String s = "BreakingBad";
out.println(s.charAt(4));
```

QUESTION 6

What is output by the code to the right?

A. `aead` B. `abcd`

C. `abeb` D. `cecd`

E. There is no output.

```
char [] list1 = {'a','b','c','d'};
char [] list2 = list1;
list2[2] = 'e';
list1[3] = list2[1];
for(char a:list1)
     out.print(a);
```

QUESTION 7

What is output by the code to the right?

A. `false false` B. `false true`

C. `true false` D. `true true`

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

```
boolean p = true;
boolean q = true;
p = p^q;
out.println(p + " " + q);
```

QUESTION 8

What is output by the code to the right?

A. `yum` B. `yumyom`

C. `burp` D. `chomp`

E. `yumyomchompburp`

```
String s1 = "sweet";
switch(s1)
{
  case "sweet":out.print("yum");
  case "sour" :out.print("yom");
               break;
  case "spicy":out.print("chomp");
  default     :out.print("burp");
}
```

QUESTION 9

What is output by the code to the right?

A. 3.1 B. 5.2 C. 8.3

D. 2.1 E. 2.5

```
out.println(Math.max(5.2,3.1));
```

QUESTION 10

What is output by the code to the right?

A. 999 B. 333

C. 342 D. 101010

E. 231

```
int[][]grid={{1,2,3},{4,5,6,7},
             {8,9}};
out.println(grid[0].length + "" +
            grid[1].length + "" +
            grid[2].length);
```

QUESTION 11

Which of the following correctly replaces **<statement1>** in the Guitar class definition on the right ?

A. public void

B. public int

C. private void

D. private int

E. public static int

QUESTION 12

Which of the following correctly replaces **<statement2>** in the Guitar class definition on the right ?

A. ( );

B. (int n)

C. ( )

D. (String s)

E. (int n);

QUESTION 13

Which of the following correctly replaces **<statement3>** in the Guitar class definition on the right ?

A. type = s;

B. numStrings = n;

C. return type;

D. return numStrings;

E. return 6;

```
class Guitar
{
  private String type;
  private int numStrings;
  public Guitar()
  {
     type = "acoustic";
     numStrings = 6;
  }
  public Guitar(int n)
  {
     this();
     numStrings = n;
  }
  public Guitar(int n, String s)
  {
     this(n);
     type = s;
  }
  public String toString()
  {
     return type + ": " +
        numStrings + " string";
  }
<statement1>getNumStrings<statement2>
  {
          <statement3>
  }
}

////////////////////////////////
////client code
Guitar g = new Guitar(5,"bass");
out.println(g);
```

QUESTION 14

What is output by the code to the right?

A. 7 B. 9

C. 15 D. 12

E. 31

```
int d = 25;
d = d | 15 & 7;
out.println(d);
```

QUESTION 15

What is output by the code to the right?

A. 99 B. 8 C. 7

D. 100 E. 0

```
int e = 0, f = 1;
while(f<100){
  e++;
  f*=2;
}
out.println(e);
```

QUESTION 16

Which term best describes the variable type for **a** in the client code shown?

A. actual parameter

B. formal parameter

C. instance field

D. class variable

E. temporary variable

QUESTION 17

What is output by the client code to the right?

A. 30 1 7

B. 30 1 1 7

C. 30 1 6 1 9 7

D. 30 1 6 45 1 9 35 1 7

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

```
static void stuff(int x)
{
  if(x%2==0)
     out.print(x*5+" ");
  else
  if(x%3==0)
     out.print(x/5+" ");
  else
     out.print(x+" ");
}
//client code
int a = 6;
stuff(a);
a+=3;
stuff(a);
a-=2;
stuff(a);
```

QUESTION 18

Which of these statements will return the substring "Probe"?

I. `s.substring(7,12);`
II. `s.substring(8);`
III. `s.substring(8,13);`
IV. `s.substring(7,13);`
V. `s.substring(7);`

A. I and V only
B. II only
C. III only
D. II and III only
E. IV only

```
String s = "Cassini_Probe";
```

QUESTION 19

What is output by the code to the right?

A. 2.9 B. 4.5 C. 19.5 D. 4.9 E. 3.2

```
long k = 12;
int m = 5;
double p = 2.5;
out.println(p+k/m);
```

QUESTION 20

What is output by the code to the right?

A. 001 010 101 111

B. 000 010 100 111

C. 001 010 101 110

D. 001 011 100 110

E. 000 010 101 111

```
for(int p = 0; p <= 1; p++)
 for(int q = 0; q <= 1; q++)
  out.print(""+p+q+(p|q&p)+" ");
```

QUESTION 21

Based on the value of x in the code on the right, which of the following statements will output only the value 6 ?

I. `out.println(x%1000/100);`
II. `out.println(x/100%10);`
III. `out.println(x/1000%10);`

A. I only
B. II only
C. III only
D. I and II only
E. I and II and III

```
int x = 49627;
```

QUESTION 22

What is output by the code to the right?

A. `360.0` B. `180.0` C. `90.0`
D. `45.0` E. `0.0`

```
double d = Math.toDegrees(Math.PI*2);
out.printf("%.1f\n",d);
```

QUESTION 23

What is output by the code to the right?

A. `2147483647`
B. `-2147483648`
C. `11110000000000000000000000000000`(4 1s, 32 zeroes)
D. `11111111111111111111111111111111` (32 1s)
E. `1111`

```
int x = 15 << 32;
String s = Integer.toBinaryString(x);
out.println(s);
```

QUESTION 24

What is output by the code to the right?

A. `true0.0`
B. `true2.3`
C. `true3.1`
D. `false2.3`
E. `false4.2`

```
ArrayList <Double> list;
list = new ArrayList<Double>();
out.print(list.isEmpty());
list.add(2.3);
list.set(0,4.2);
list.add(3.1);
list.remove(0);
out.print(list.get(0));
```

QUESTION 25

Find f(12,6) according to the recursive function definition shown on the right. You may use the space below to do your work.

f(12,6) =

$$f(x,y) = \begin{cases} f(x-y,y-1)+2 & \text{when } x>y \\ x+y & \text{otherwise} \end{cases}$$

A. `5` B. `6` C. `7`
D. `9` E. `12`

QUESTION 26

What is output by the code to the right?

A. `Fry` B. `FryFa`

C. `FreettyFall` D. `FreeFallinTomPetty`

E. There is no output due to a compile error

```
String s = "FreeFallinTomPetty";
String [] ar = s.split("[elt]+");
out.println(ar[0]+ar[ar.length-1]
          +ar[1]);
```

QUESTION 27

What is output by the code to the right?

A. `1` B. `33`

C. `100` D. `bad`

E. `breaking`

```
String bb = (100%3==0)?"breaking"
                      :"bad";
out.println(bb);
```

QUESTION 28

What is output by the code to the right?

A. `false` B. `-8`

C. `8` D. `-1`

E. `1`

```
String s = "KarelJRobot";
String t = "Kilamanjaro";
out.println(s.compareTo(t));
```

QUESTION 29

A. `10` B. `20`

C. `ten` D. `sepuluh`

E. `tensepuluh`

```
Map<Integer,String> m =
  new HashMap<Integer,String>();
m.put(10,"ten");
m.put(14,"fourteen");
m.put(9,"nine");
m.put(10,"sepuluh");
out.println(m.get(10));
```

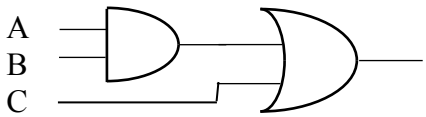
QUESTION 30

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

A. A && B || C B. A || B && C

C. A ^ B || C D. A || B ^ C

E. A && B ^ C



QUESTION 31

On the right is a boolean expression using generic notation. Which of the expressions below represents the simplest form of this expression ? (Note : * means AND, + means OR)

A. $\overline{A}$ B. 0 C. $\overline{A} * \overline{B}$ D. $\overline{A}(\overline{A}*\overline{B})$ E. $\overline{A}+\overline{B}$

$$\overline{A}\,\overline{(A + B)}$$

(this translates to ***"not A and not (A or B)"***)

QUESTION 32

In a typical binary search process, in how many steps will the value 5 be found in the array shown on the right?

A. 3 B. 4

C. 5 D. 6

E. 7

```
0 1 2 3 4 5 6 7 8 9 10 11 12 13
```

QUESTION 33

Which statement below best describes the minimum required <implementation> of class B for the class structure shown on the right?

A. class B is only required to define method **`one()`**.
B. class B is not required to implement anything.
C. class B is required to implement method **`one()`** and override method **`two()`**.
D. class B is only required to override method **`two()`**.
E. This class structure is invalid.

QUESTION 34

Suppose all is correctly defined with this class structure so that method one()returns the value 4. What is the output for the client code shown on the right?

A. `0` B. `5`

C. `20` D. `40`

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

```
abstract class A
{
  int x = 2;
  abstract int one();
  int two()
  {
     return 5;
  }
}
class B extends A
{
  //<implementation>
}

//////////client code///////////////
B bop = new B();
out.println(bop.one()*bop.two()
              *bop.x);
```

QUESTION 35

Which of the following is an **INVALID** class B definition?

I.
```
class B extends A{
      int one(){
         return 4;
      }}
```
II.
```
class B extends A{
      x=1;
      int one(){
         return 4;
      }}
```
III.
```
class B extends A{
      int one(){
         return 4;
      }
      int two(){
         return 6;
      }}
```
IV.
```
class B extends A{
      int x = 4;
      int one(){
         return 4;
      }
      int two(){
         return 6;
      }}
```

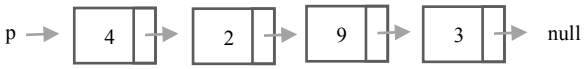
A. I is invalid
B. II is invalid
C. III is invalid
D. IV is invalid
E. All of these are valid

QUESTION 36

Suppose a linked list has been implemented as shown in the diagram on the right, with public fields **`data`** and **`next`**. What is the output of the statement below?

```
out.print(p.next.data);
```

A. 2 B. 3 C. 4 D. 5 E. 9



QUESTION 37

What is output by the code to the right?

A. 3null

B. 3false

C. 3true

D. 4false

E. 4true

```
Set<Integer> sa = new
  TreeSet<Integer>();
sa.add(4);
sa.add(5);
sa.add(4);
sa.add(6);
sa.add(7);
sa.remove(6);
out.print(sa.size());
out.println(sa.contains(6));
```

QUESTION 38

What is the output of this code if the value of ***<keyboard integer input>*** is 3.14?

A. Bad data.

B. All is good.

C. Bad data. All is good.

D. There is no output.

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

```
int tx;
try{
  tx = <keyboard integer input>;
}
catch(Exception ee){
  out.print("Bad data. ");
}
finally{
  out.print("All is good. ");
}
```

QUESTION 39

On the right is a binary tree implementing a min heap, with the 1 in position 0, the 7 in position 1, and the 5 in position 2. The last element added was a 2. In what position does the value 2settle when the min heap is reestablished in the sifting up process?

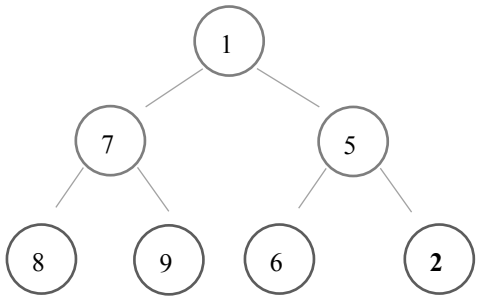
A. position 0

B. position 1

C. position 2

D. position 5

E. position 6

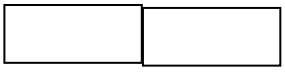


QUESTION 40

OPEN ENDED QUESTION – Using the ***enqueue*** and ***dequeue*** sequence given on the right, process the commands shown into a standard queue and indicate the ***last value dequeued*** and which value would be the ***next one dequeued***.

*Find the **two** answers and write them on your answer sheet **correctly labeled**. If using a ScanTron form, write them out to the side of the bubbles, also **correctly labeled**. If not labeled, the order you put your answers will be assumed to be **last value dequeued**, then **next value to be dequeued**.*

Last value dequeued Next value to be dequeued



```
enqueue 3
enqueue 5
enqueue 4
dequeue x
enqueue 7
dequeue x
dequeue x
enqueue 9
```

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 2014

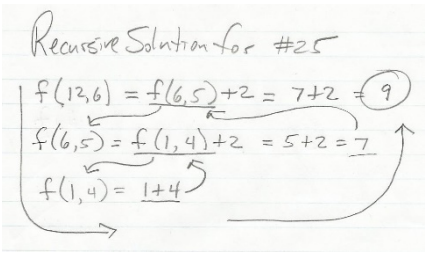
1) D	11) B	21) D	31) C
2) B	12) C	22) A	32) B
3) D	13) D	23) E	33) A
4) A	14) E	24) C	34) D
5) E	15) C	25) D	35) E
6) C	16) A	26) B	36) A
7) B	17) A	27) D	37) B
8) B	18) D	28) B	38) C
9) B	19) B	29) D	39) C
10) C	20) E	30) A	40) 4 last value popped 7 next to be popped

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.

Brief Explanations:

1. $11110_2 + 11011_2 = 30_{10} + 27_{10} = 57_{10} = 71_8 = 39_{16} = 111001_2$
2. b = 19 + 13 = 32, c = 13
3. Integer object array cannot be initialized with a double
4. j starts at 5, outputs from 4 down to 1, and stops at 1
5. character at position 4 is k
6. list2 is an alias for list 1, so any changes made by one are changes made to the other
7. p^q is p xor q, which requires opposites in order to be true. Since both are true, the result for p is false.
8. This string switch statement matches at "sweet", and outputs both "yum" and "yom" since the break is only after "yom"
9. The maximum of 5.2 and 3.1 is 5.2
10. The lengths of each row of this uneven grid are 3, 4, and 2
11. getNumStrings is an accessor method with a heading of public int since it returns an integer
12. and requires no parameter, so has empty ( ) with NO semicolon!
13. and simply returns the numStrings instance field value
14. Since & evaluates first in bitwise order, 15&7 results in 7, and then 25 | 7 results in 31.
15. This is a loop that calculates the log base 2 of 100…e increments by 1, but f doubles each time and passes 100 at the 7th iteration
16. The parameter in a method call is called the actual parameter
17. Since this is a chain if else, only one value is output for each call, according to the logic of the if statements. 6 produces 30, 9 produces 1, and 7 produces 7.
18. Both the (8) and (8,13) substring calls produce the word "Probe" from this string since the P is at position 8 and 13 is the length of the string, one step PAST the end of the substring desired.
19. This expression follows the order of operations and integer divides 12 and 5 to get 2, then adds 2.5 to get 4.5
20. The Boolean expression is p or q and q, which when simplified just becomes p (Law of Absorption) and therefore each output digit matches the p digit of the term.
21. Both I and II options correctly (but in different ways) isolate the 6. Option III isolates the 9.
22. 2PI is just a full circle..360 degrees.
23. Any integer left shifted 32 spots (the bit size of the integer data type), will simply return to its original value. Essentially it is a Left Circle back to the original number.
24. The list is indeed empty to start with, and after all the action, 3.1 is the sole surviving element in this list at position 0.
25. See the recursive trace on the right for the solution to this problem.



26. The "[elt]+" pattern splits at any sequence of the letters e, l and t, which produces the array ["Fr","Fa", "inTomP","y"] in this instance, producing "FryFa" for this output.
27. This ternary operation results in false, since 100%3 is not 0, therefore the resulting string is the one following the :, which is "bad", **very "bad"**!!!
28. The first different characters in these two strings are 'a' and 'i', which produces -8 since 'a' is 8 places before 'i'.
29. Since the value 10 can only map to one value, the "ten" is replaced with "sepuluh", another word for 10. Look it up!
30. Another digital electronics question! Don't you just love 'em? Just learn the basic symbols and this will become very easy for you. The bullet shape is the AND, and the arrow is OR. This is simply A and B or C.
31. DeMorgan's law is applied to the **not(a or b)**, resulting in **not a and not b**, which when "anded" with another **not a** simply becomes **not a and not b**.
32. Binary search is easy. Find the middle, and if doesn't match, go left or right and find the middle again. Repeat this process until the middle is the one you want. Then count the "middles" and that's how many steps you took to find it.
33. This structure is most certainly valid. Any class inheriting an abstract class is required to implement any abstract method in that class, so class B is REQUIRED to implement method one() from class A. Anything else is optional.
34. This is just simply 4 from method one times 5 from method two times the 2 from the variable x, for a result of 40.
35. All of these implementations of class B are valid. Look them over carefully.
36. Since p.next pointer references the second node of the list, and the data for that node is 2, the resulting output is 2.
37. In this TreeSet process, 4 is added twice, but since there are no duplicates in sets, only remains once. The 6 is removed, leaving only the 4, 5, and 7, so the size is 3 and 6 is not in the list.
38. Data input is a classic use of the try catch block. Since 3.14 is a mismatch for integers, the exception is thrown by the try block and caught by the catch block, resulting in the "Bad data. " output. The finally block ALWAYS occurs, no matter what.
39. In the heapify process of a min heap, the process always starts at the bottom right of the tree, working left and upwards, switching any parent and child values that are not in correct min heap order. The first such occurrence here is the 6 and 3. Next will be the 1 and 2, and so on.
40. In this queue push and pop sequence, the 3,5, and 4 are pushed, then the 3 is popped, push the 7, pop the 5, pop the 4, then push the 9. The 4 was the last value popped, and the 7 sits at the front of the queue, waiting to be popped next.