University Interscholastic League
Computer Science Competition

Number 144 (Invitational B - 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 $110_2 + 100010_2$?

A. 40_{10} B. 46_8 C. 28_{16} D. 101000_2 E. All are equivalent

QUESTION 2

What is output by the code to the right?

A. 4 B. 4.8

C. 5 D. 5.0

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

```
int h = 24;
h/=5;
out.println(h);
```

QUESTION 3

What is output by the code to the right?

A. 3 B. 4 C. 4.0

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

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

```
Double [] list = {1.0,2.0,3.0,4};
out.println(list[3]);
```

QUESTION 4

What is output by the code to the right?

A. 369 B. 36912

C. 6912 D. infinite loop

E. There is no output.

```
int k = 3;
do
{
  k+=3;
  out.print(k);
}
while (k!=12);
```

QUESTION 5

What is output by the code to the right?

A. 0 B. 1 C. 5 D. 6 E. 7

```
String s = "beachbum";
out.println(s.indexOf(98,1));
```

QUESTION 6

What is output by the code to the right?

A. 0.12.34.56.7 B. 2.36.74.54.5

C. 4.54.54.54.5 D. 6.74.52.30.1

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

```
double [] list = {0.1,2.3,4.5,6.7};
list[3]=list[2];
list[1]=list[3];
list[0]=list[1];
for(double d:list)
  out.printf("%.1f",d);
```

QUESTION 7

For which initial values of p and q will this expression output true?

I. p=false;q=false
II. p=false;q=true
III. p=true;q=false;
IV. p=true;q=true ;

A. I and IV only
B. II and III only
C. IV only
D. I, II, and III only
E. All will work.
F.

```
boolean p = ?;
boolean q = ?;
out.println(p^q);
```

QUESTION 8

For which of these inputs will the final value of sum be greater than zero?

A. `"a"`
B. `"aa"`
C. `"bb"`
D. `"cccc"`
E. `""`

```
String s = <string value>;
int sum = 0;
switch(s)
{
  case "a"    : sum += s.length();
  case "bb"   : sum -= s.length();
  case "cccc" : sum *= -s.length();
  case ""     : sum--;
}
out.println(sum);
```

QUESTION 9

What is output by the code to the right?

A. `3.1`
B. `5.2`
C. `-5.2`
D. `-2.1`
E. `-3.1`

```
out.println(Math.min(-5.2,3.1));
```

QUESTION 10

Which statement will correctly output the value 6 from the array shown to the right?

A. `out.print(a[1][2]);`
B. `out.print(a[2][3]);`
C. `out.print(a[5]);`
D. `out.print(a[2][1]);`
E. `out.print(a[3][2]);`

```
int[][]a={{1,2,3},{4,5,6,7},{8,9}};
```

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>setNumStrings<statement2>
  {
           <statement3>
  }
}
```

QUESTION 14

What is output by the code to the right?

A. 0 B. 15
C. 25 D. 34
E. 1073741823

```
int d = 30;
d = d ^ 15 << 1;
out.println(d);
```

QUESTION 15

What is output by the code to the right?

A. 0 B. 39 C. 40

D. 120 E. 121

```
int j = 0;
do{
  j+=2*j;
  j++;
}
while(j<50);
out.println(j);
```

QUESTION 16

Which term best describes the method type in the code shown to the right?

I. static method
II. void method
III. return method
IV. mutator method

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

QUESTION 17

What is output by the client code to the right?

A. -5593680105
B. 439
C. 651
D. 535
E. 9814

```
static int stuff(int x)
{
  if(x%9>5)
     return (x%9-5);
  if(x%9<5)
     return (x%9+5);
  return (x%9);
}

//client code
out.print(stuff(9));
out.print(stuff(8));
out.print(stuff(14));
```

QUESTION 18

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

A. s.substring(6);
B. s.substring(7);
C. s.substring(6,6);
D. s.substring(6,7);
E. s.substring(7,8);
F.

```
String s = "FenderRumble";
```

QUESTION 19

What is output by the code to the right?

A. -2 B. -3 C. 21 D. 22 E. 25

```
int d = 9;
int f = 60;
int g = 31;
out.println(g-f%d);
```

QUESTION 20

What is output by the code to the right?

A. `000 010 101 111`

B. `000 011 100 111`

C. `001 010 101 110`

D. `001 011 101 110`

E. `000 010 101 110`

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

QUESTION 21

What is output by the code to the right?

A. `1` B. `2` C. `3` D. `1.0` E. `1.5`

```
double g = 28.5;
out.println(g%9);
```

QUESTION 22

What is output by the code to the right?

A. `0.79` B. `1.05` C. `1.57` D. `3.14` E. `6.28`

```
d = Math.toRadians(180.0);
out.printf("%.2f\n",d);
```

QUESTION 23

What is output by the code to the right?

A. `2147483644`

B. `-2147483645`

C. `11001111111111111111111111111111` (32 digits)

D. `11000000000000000000000000000000` (32 digits)

E. `1100`

```
int x = 12 << 32;
String s = Integer.toBinaryString(x);
out.println(s);
```

QUESTION 24

What is output by the code to the right?

A. `4 null`

B. `4 6`

C. `5 null`

D. `5 6`

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

```
ArrayList lost = new ArrayList(5);
lost.add(null);
lost.add(new Integer(6));
lost.add("ball");
lost.add(4.7);
out.println(lost.size()+"
"+lost.get(1));
```

QUESTION 25

Find f(10,5) according to the recursive function definition shown on the right. You may use the space below to do your work.

f(10,5) =

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

A. `5` B. `6` C. `7`

D. `8` E. `10`

QUESTION 26

What is output by the code to the right?

A. `il` B. `vain`

C. `ilovetopaint`

D. There is no output due to a runtime error

E. There is no output due to a compile error

```
String s = "ilovetopaint";
String [] ar = s.split("[pote]");
out.println(ar[1]+ar[5]);
```

QUESTION 27

What is output by the code to the right?

A. `0` B. `5`

C. `100` D. `dead`

E. `walking`

```
String b = (100%5==0)?"walking"
                     :"dead";
out.println(b);
```

QUESTION 28

What is output by the code to the right?

A. `-1` B. `1`

C. `-15` D. `15`

E. `false`

```
s = "SperryRand";
t = "SpecialK";
out.println(s.compareTo(t));
```

QUESTION 29

A. `nine` B. `9`

C. `ten` D. `sepuluh`

E. `null`

```
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(0));
```

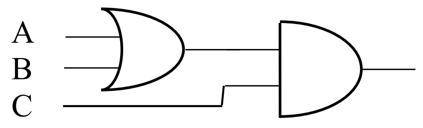
QUESTION 30

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, ⊕ means XOR)

$(A \oplus B)\,(A + B)$

(this translates to ***"A xor B and A or B)"***)

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

D. False E. A+B

QUESTION 32

In a typical binary search process, in how many steps will the value 8 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 **`two()`**.
B. class B is not required to implement anything.
C. class B is required to implement method **`two()`** and override method **`one()`**.
D. class B is only required to override method **`one()`**.
E. This class structure is invalid.

QUESTION 34

Suppose all is correctly defined with this class structure so that method **two()** returns the value 2. 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.

QUESTION 35

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

I.
```
class B extends A{
      int two(){
         return 2;
      }}
```
II.
```
class B implements A{
      x=1;
      int two(){
         return 2;
      }}
```
III.
```
class B extends A{
      int one(){
         return 5;
      }
      int two(){
         return 2;
      }}
```
IV.
```
class B extends A{
      int x = 4;
      int one(){
         return 5;
      }
      int two(){
         return 2;
      }}
```

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

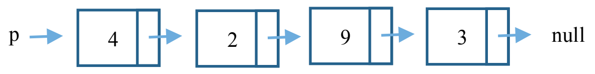
```
abstract class A
{
  int x = 2;
  int one()
  {
     return 5;
  }
  abstract int two();

}
class B extends A
{
  //<implementation>
}

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

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.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(4);
out.print(sa.size());
out.println(sa.contains(6));
```

QUESTION 38

What is the output of this code if the value of ***`<keyboard 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.

```
double tx;
try{
  tx = <keyboard 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 max heap, with the 9 in position 0, the 7 in position 1, and the 5 in position 2. The last element added was a 12. In what position does the value 12 settle when the min heap is reestablished in the sifting up process?

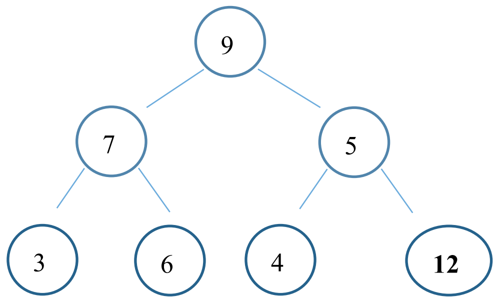
A. position 0

B. position 1

C. position 2

D. position 5

E. position 6



QUESTION 40

OPEN ENDED QUESTION – Using the generic push and pop sequence given on the right (**push** to mean Java's ***enqueue***, **pop** to mean Java's ***dequeue***), process the commands shown on the right into a queue and indicate the ***last value popped*** and which value would be the ***next one popped***.

*Find the **two** answers and write them on your answer sheet **correctly labeled**. If using a ScanTron form, 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 popped**, then **next value to be popped**.*

Last value popped	Next value to be popped

Push 9
Push 7
Pop x
Push 5
Push 8
Push 6
Pop x
Pop x

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 B 2014

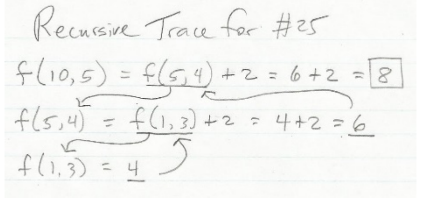
1) B	11) A	21) E	31) B
2) A	12) B	22) D	32) A
3) D	13) B	23) E	33) A
4) C	14) A	24) B	34) C
5) C	15) E	25) D	35) B
6) C	16) D	26) B	36) E
7) B	17) D	27) E	37) C
8) C	18) D	28) D	38) B
9) C	19) E	29) E	39) A
10) A	20) B	30) D	40) 5 last value popped 8 next to be popped

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. $110_2 + 100010_2 = 6_{10} + 34_{10} = 40_{10} = 50_8 = 28_{16} = 101000_2$
2. H = 24/5 = 4 (integer division)
3. Even though this is an array of Double objects, autoboxing **does not apply** when instantiating a static array like this. The 4 is an int and will cause a compile error, "incompatible types"
4. k starts at 3, outputs 6, 9 and 12, and stops at 12
5. the character 98 is the letter 'b', at position 5 from position 1. The 'b' in position 0 is not considered.
6. By the end of this assignment sequence, 4.5 is the element in every position.
7. p^q is p xor q, which requires opposites in order to be true, therefore p=true;p=false or p=false;q=true; will both evaluate to true.
8. The resulting values for all five choices are: "a",-1,"aa",0,"bb",3,"cccc",-1,"",-1
9. The minimum of -5.2 and 3.1 is -5.2
10. The 6 is in the second row (row 1), and in the third position of that row (column 2).
11. **`setNumStrings`** is a mutator method with a heading of **`public void.`**
12. It receives an integer **`(int n)`**
13. and assigns it to numStrings (**`numStrings = n;`**)
14. Since shift operations have priority over bitwise operations, 15 is left-shifted first, becoming 30, then 30 xor 30 is zero (Any value xor itself is zero – in assembly language that is one way of assigning a value of zero to a register).
15. The j values in this loop sequence are: 0, 0, 1, 3, 4, 12, 13, 39, 40, 120 and finally 121.
16. As is evident in the heading, this method is both a static method and a return method.
17. Since this is a chain if else, only one value is output for each call, according to the logic of the if statements. 9 produces 5, 8 produces 3, and 14 produces 5.
18. The (6,7) substring call is the correct one to access the letter "R".
19. This expression follows the order of operations, where 60%9 produces 6, and then is subtracted from 31 to make 25.
20. The Boolean expression is **p and q or q**, which when simplified just becomes **q** (Law of Absorption) and therefore each output digit matches the **q** digit of the term.
21. 28.5 mod 9 produces the value 1.5.
22. 180 degrees in radians is PI.
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. The Integer.toBinaryString method only outputs significant digits…no leading zeroes.
24. Since this the type of this ArrayList is not designated, any mixture of objects, including null, is acceptable. The final contents of the array are [null, 6, "ball", and 4.7].
25. See the recursive trace on the right for the solution to this problem.



Recursive Trace for #25

$f(10,5) = \underline{f(5,4)} + 2 = 6+2 = \boxed{8}$

$f(5,4) = \underline{f(1,3)} + 2 = 4+2 = \underline{6}$

$f(1,3) = \underline{4}$

26. The `"[pote]"` pattern splits at any of the letters 'p', 'o', 't' and 'e', which produces the array `["il","v","","","","ain"]` in this instance, producing `"vain"` for this output.
27. This ternary operation results in true, since 100%5 is equal to 0, therefore the resulting string is the one following the ?, which is `"walking"`.
28. The first different characters in these two strings are 'r' and 'c', which produces 15 since 'r' is 15 places after 'c'.
29. Since hash structures guarantee no certain order, there is no indexing, therefore the call get(0) does NOT return the first element of the hash mapping, but simply looks for the mapping of the key value zero, and finding none, returns null.
30. Since the OR happens before the AND in this case, and AND occurs before OR in logic order, it necessary to use parentheses, producing (A OR B) AND C.
31. $A \oplus B$ simplifies to (not A and B or A and not B), which when FOILED with (A+B) produces the same thing, $A \oplus B$.
32. The first "middle" found is the 6. Since 7 is to the right, the next "middle" is 9, then going left where the final "middle" is 7, the search item.
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 two() from class A. Anything else is optional.
34. This is the product of 5 from method one, 2 from method two, and 2 from the variable x, for a result of 20.
35. The word "implements" is used when an interface is used, therefore option II is not valid. All of the rest are valid.
36. Since p.next.next pointer references the third node of the list, and the data for that node is 9, the resulting output is 9.
37. In this TreeSet process, 4 is added twice, but since there are no duplicates in sets, only instance remains. When the 4 is removed, leaving only the 5, 6, and 7, the size of the set is 3 and 6 is indeed in the list, resulting in the output **3true**.
38. Data input is a classic use of the try catch block. Since 3.14 works for doubles, no exception is thrown, and program flow drops to the finally block, which always executes, no matter what.
39. In the heapify process of a max 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 max heap order. The first such occurrence here is the 6 and 7. Next will be the 2 and 9, and so on.
40. In this queue push and pop sequence, the 9 and 7 are pushed, then the 9 is popped, push the 5, 8, and 6, pop the 7 and the 5. The 5 was the last value popped, and the 8 sits at the front of the queue, waiting to be popped next.