University Interscholastic League

COMPUTER SCIENCE WRITTEN TEST STUDY PACKET 2015-2016

A DOWNLOADABLE VERSION OF THE PACKET THAT INCLUDES PROGRAMMING MATERIALS IS AVAILABLE FOR PURCHASE THROUGH THE UIL ONLINE STORE:

http://store.uiltexas.org

VISIT THE UIL WEBSITE FOR LINKS TO ADDITIONAL COMPUTER SCIENCE RESOURCES AND STUDY AND PRACTICE MATERIAL:

http://www.uiltexas.org/academics/computer-science

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`

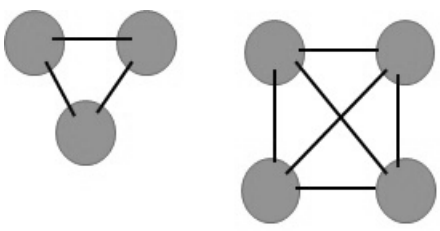
C. `00000000` D. `14.00000`

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?

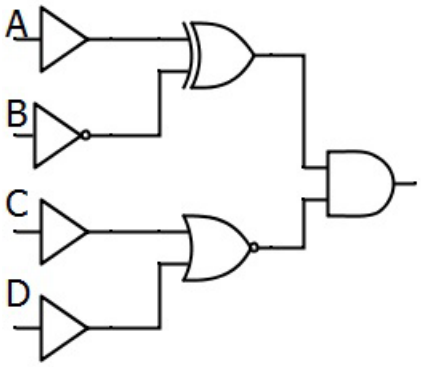
A. $A \oplus \overline{B} * \overline{C+D}$

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 Sheet

Conference _____

Code Number _____

1. _____	11. _____	21. _____	31. _____
2. _____	12. _____	22. _____	32. _____
3. _____	13. _____	23. _____	33. _____
4. _____	14. _____	24. _____	34. _____
5. _____	15. _____	25. _____	35. _____
6. _____	16. _____	26. _____	36. _____
7. _____	17. _____	27. _____	37. _____
8. _____	18. _____	28. _____	38. _____
9. _____	19. _____	29. _____	39. _____
10. _____	20. _____	30. _____	40. _____

FOR GRADING USE ONLY

Number Correct _____ **x 6 =** _____

Number Incorrect _____ **x 2 =** _____

Grader 1 Initial _____

Grader 2 Initial _____

Grader 3 Initial _____

Subtract Line 2 above from Line 1.

SCORE

_____

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.

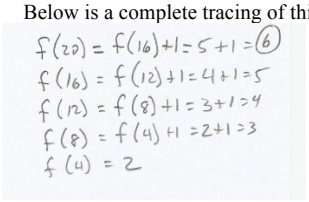
Explanations:

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

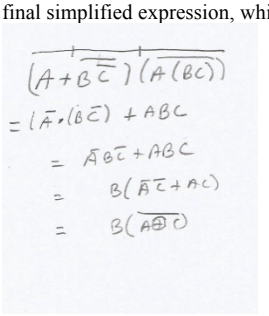
34. C The first rule is that the operands stay in the same order, 3 9 6 5 2. Then insert the operators as shown. The * is immediately after the 3 and 9, so it goes between them, likewise with the ^ between the 5 and 2. The - is after the 3*9 and 6, so it goes between them, and finally the + goes between 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.



University Interscholastic League
Computer Science Competition

Number 150 (Invitational B - 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 $10111000_2 + AB_{16}$?

A. 163_{16} B. 355_{10} C. 543_8 D. 10110001_2 E. All are equivalent

QUESTION 2

What is the result of the expression shown?

A. 1 B. 0.56

C. 2 D. -1 E. 0

```
23 / 9 * 1 % 2 = _____
```

QUESTION 3

What is output by the code to the right?

A.
```
----*----*
    49.2
```
B.
```
----*----*
49.20
```
C.
```
----*----*
9.20
```
D.
```
----*----*
49.2
```
E.
```
----*----*
  49.20
```

```
System.out.println("----*----*");
System.out.printf("%4.2f",49.2);
```

QUESTION 4

What is output by the code to the right?

A. UILCOMPUTERSCIENCE2015

B. UOLCOMPUTORSCOONCO2015

C. OUOOLOCOOOMOPOUOTOOROSOCOONOCOO2O0O1O5O

D. UOLCOMPUTORSCONCO2015

E. UILCOMPUTERSCONCE2015

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

QUESTION 5

What is output by the code to the right?

A. false B. true

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

QUESTION 6

What is output by the code to the right?

A. 17.0 B. 11.5 C. 23.0 D. 12.7

E. There is no output due to an error.

```
int x = 8;
double y = 15;
out.println(Math.hypot(8,15));
```

QUESTION 7

What is output by the code to the right?

A. 11 B. 11.2 C. 8.0 D. 8

E. There is no output due to an error.

```
int i = 8;
double d = 1.4;
out.printf("%d",i*=d);
```

QUESTION 8

What is output by the code to the right?

A. 16 B. 160

C. 14 D. 40

E. 4

```
int x = 14,y=0;
switch(x%3)
{
 case 0:y+=0;break;
 case 1:y+=1;
 case 2:y+=2;break;
 case 3:y+=3;
 default:y*=10;
}
out.println(x+y);
```

QUESTION 9

What is output by the code to the right?

A. 78 B. 159 C. 128

D. 79 E. There is no output due to an error.

```
int x = 4;
for(;x<75;x++)
 x*=2;
out.println(x);
```

QUESTION 10

What is output by the code to the right?

A. 0 7 4 5 3 6 2 0 B. 0 6 3 4 2 5 1 0

C. 6 3 4 2 5 1 0 0 D. 7 4 5 3 6 2 0 0

E. There is no output due to an error.

```
int a=1;
int[]list1={5,3,1,2,4,0};
int[]list2=new int[8];
for(int x:list1)
 list2[x+1]=++a;
for(int x:list2)
  out.print(x+" ");
```

QUESTION 11

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

```
154
```

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

A. `String n = f.nextLine();`

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

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

D. `String n = f.next();`

E. All code segments will work properly

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

QUESTION 12

What is output by the code to the right?

A. 8

B. 7

C. 6

D. 5

E. 4

```
double d = 100;
int x = 0;
while(d<1000)
{
 d*=1.5;
 x++;
}
out.println(x);
```

QUESTION 13

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

I. `||`
II. `|`
III. *`expr++ expr--`*

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

QUESTION 14

The integer data type **byte** uses 8 bits of storage, which means it has 2^8, or 256 possible values. Which of the choices below indicates the range of values for this data type.

A. -128 to 128 B. -128 to 127 C. 1 to 256 D. 0 to 255 E. -127 to 128

QUESTION 15

The output for the code to the right is: `0 1 2 0 1 2 0 1 2`

Which choice replaces **<statement>** in the code to the right so that it compiles and runs correctly?

A. `double`
B. `Object`
C. `Integer`
D. `String`
E. `int`

```
Integer [] list={0,1,2};
ArrayList<Integer> aList = new
ArrayList<Integer>();
for(Integer x:list)
 aList.add(x);
<statement> [] list2 =
aList.toArray();
for(Integer x:list)
 out.print(x+" ");
for(Integer x:aList)
 out.print(x+" ");
for(<statement> x:list2)
 out.print(x+" ");
```

QUESTION 16

How many ordered triples make this boolean expression false?

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

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

QUESTION 17

What is output by the code to the right?

A. `-96` B. `78` C. `96` D. `-78`
E. There is no output due to an error.

```
char x = 'A';
char y = 'a';
int z = 3;
out.println((x-y)*z);
```

QUESTION 18

What is output by the code to the right?

A. `0 24 9 0 39`
B. `0 36 21 6 0`
C. `0 12 0 42 27`
D. `0 48 33 18 3`
E. `0 0 45 30 15`

```
int[][]nums = new int[5][5];
for(int x = 3;x<50;x+=3)
 nums[x%5][x%4+1]=x;
for(int x = 0;x<5;x++)
 out.print(nums[2][x]+" ");
out.println();
```

QUESTION 19

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 8-bit binary representation of -118?

A. 10001100 B. 10001110 C. 10001010 D. 10001011 E. 10001101

QUESTION 20

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

A. O(N^2)

B. O(N)

C. O(1)

D. O(N log N)

E. O(log N)

QUESTION 21

What is output by the client code to the right?

A. 3 4 5 6 7 9 B. 5 7 3 9 4 6

C. 9 7 6 5 4 3 D. 6 4 9 3 7 5

E. Not possible to determine

QUESTION 22

Which of the six indicated <statements> in the code to the right need to be altered in order to reverse the sorting order?

I. <statement 1>
II. <statement 2>
III. <statement 3>
IV. <statement 4>
V. <statement 5>
VI. <statement 6>

A. II only

B. III and IV only

C. III only

D. IV, V, and VI only

E. I only

```
public static void quickSort (int a[], int
lo, int hi){
  <statement 1>
  if (lo >= hi) return;
  int left = lo;
  int right = hi;
  <statement 2>
  int pivot = a[(lo+hi)/2];

  while ( left < right)  {
    <statement 3>
    while (a[left] > pivot) left++;
    <statement 4>
    while (pivot > a[right]) right--;
    <statement 5>
    if (left <= right)    {
      swap (a, left, right);
      left++;
      right--;
    }
  }
  <statement 6>
  quickSort (a, lo, right);
  quickSort (a, left, hi);
}

public static void swap (int a[],
                         int i, int j){
  int tmp = a[i];
  a[i] = a[j];
  a[j] = tmp;
}

public static void outputList(int[]list)
{
 for(int x=0;x<list.length;x++)
    out.print(list[x]+" ");
 out.println();
}
//client code
 int[]list = {5,7,3,9,4,6};
 quickSort(list);
 outputList(list);
```

QUESTION 23

How many 'o's will be output in the code to the right?

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

E. There is no output due to an error.

```
String s =
"abcdefghijklmnopqrstuvwxyz";
char [] list = s.toCharArray();
char a = list[0];
while(a++!='z')
 for(char b:list)
    if("aeiou".indexOf(b,a-98)>=0)
       out.print(b);
```

QUESTION 24

What is output by the code to the right?

A. 1.7 B. 0.9 C. 0.5

D. 1.0 E. None of these

```
int angle = 45;
out.printf("%.1f\n",
    Math.tan(Math.toRadians(angle)));
```

QUESTION 25

What is output by the code to the right?

A. 01111101 B. 01111010 C. 11111101

D. 00000010 E. 11111110

```
byte c = -10;
c>>=2;
out.println(Integer.toBinaryString(c)
.substring(24));
```

QUESTION 26

On the right is a fairly common version of the binary search algorithm, a standard search process used in computer science. What required process (if any) needs to replace **//line A** in the client code for this algorithm to work properly?

A. Sort the list in ascending order.

B. Reverse the order of the list.

C. There is no required process. The list is fine as is.

D. Sort the list in descending order.

E. Process the list into a hash table.

QUESTION 27

What is output by the client code to the right?

A. 0 0

B. 7 -1

C. 7 -2

D. 0 -1

E. 8 -1

```
static int binarySearch(int[] elements,
                        int target) {
 int left = 0;
 int right = elements.length - 1;
 while (left <= right) {
  int middle = (left + right) / 2;
   if (target < elements[middle]){
      right = middle - 1;
   }
   else if (target > elements[middle]){
      left = middle + 1;
   }
   else
      return middle;
 }
 return -1;
}
<client code>
int [] list = {5,-7,3,9,4,8,-3, 1,-5, 0};
//line A
out.print(binarySearch(list,5)+" ");
out.println(binarySearch(list,-6));
```

QUESTION 28

Which statement best describes the string patterns listed below as each replaces the **<pattern>** segment in the code to the right?

I. ".*" II. ".+" III. ".?"

A. All produce true outputs

B. All produce false outputs

C. II and III produce true, I produces false

D. I and II produce true, III produces false

E. I and III produce true, II produces false

```
String s = "Invitational B";
boolean p = Pattern.matches(<pattern>,s);
out.println(p);
```

QUESTION 29

What is output by the code to the right?

A. 0 -2 3 B. 6 10 -5

C. 5 11 3 D. 5 -2 3

E. There is no output due to an error.

```
 public static int myst(int x)
 {
  if(x<0)return 3;
  if(x==0)return 1;
  return myst(x-3)+2;
 }
//client code
out.print(myst(6)+" ");
out.print(myst(10)+" ");
out.println(myst(-5));
```

QUESTION 30

What is the output of the code to the right?

A. `BBBBAAAABBB`

B. `AAAABBBBAAA`

C. `AAABBBBBBAA`

D. `BBBAAAAAABB`

E. There is no output due to an error.

```
String s = "UIL 2015 CS";
char[]list=s.toCharArray();
for(char a:list)
 out.print(a>60?"A":"B");
```

QUESTION 31

What is output by the code to the right?

A. `7 28 59`

B. `7 34 63`

C. `7 42 59`

D. `7 42 77`

E. None of these

```
out.print(Integer.toOctalString(7));
out.print(Integer.toOctalString(34));
out.println(Integer.toOctalString(63));
```

QUESTION 32

Using the space provided, create a binary search tree using the letters, INVITATIONALB. After creating the tree, select the choice that shows <u>how many parent nodes have only one child</u>, and the <u>height of the resulting tree.</u>.

Assume that the initial tree shown has a height of zero, and that duplicate letters **<u>are allowed in the tree</u>**, and slide to the **left** of matching elements.

I

/ \

A. 5 6

B. 6 5

C. 6 6

D. 5 5

E. None of these

QUESTION 33

What is output by the statements in **section 1** of the client code below?

```
//section 1
ThingOne t1 = new ThingOne();
out.print(t1);
t1.setThing(6);
out.print(t1);
//section 2
t1=new ThingTwo(3);
out.print(t1.getThing());
//section 3
t1.reduceThing(1);
out.println(t1);
```

A. F5F6

B. AFDF

C. AF5DF6

D. A5D6

E. None of these

QUESTION 34

What is output by the statements in **section 2** of the client code in question 33?

A. HAI3

B. AHI3

C. AHI5

D. HI3

E. HAI5

QUESTION 35

What is output by the statements in **section 3** of the client code in question 33?

A. KLF7

B. KL52

C. L52

D. KLF52

E. KL7

```
class ThingOne{
 public ThingOne(){
  thing = 5;
  out.print("A");
}
 public ThingOne(int t){
  thing = t;
  out.print("B");
}
 public int getThing(){
  out.print("C");
  return thing;
}
 public void setThing(int t){
  thing = t;
  out.print("D");
}
 public void reduceThing(int t){
  thing -=t;
  out.print("E");
}
 public String toString(){
  out.print("F");
  return ""+thing;
}
 private int thing;
}
class ThingTwo extends ThingOne{
 public ThingTwo(){
  super();
  out.print("G");
  thing = 4;
 }
 public ThingTwo(int t){
  thing = t;
  out.print("H");
}
 public int getThing(){
  out.print("I");
  return thing;
}
 public void setThing(int t){
  thing = t;
  out.print("J");
}
 public void reduceThing(int t){
  thing -=t;
  out.print("K");
}
 public String toString(){
  out.print("L");
  return super.toString()+thing;
}
 private int thing;
}
```

QUESTION 36

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 prefix expressions below matches the infix expression shown?

`A+B-C*D+E^F`

A. **`+AB+*CD-^EF`**

B. **`+-AB-*CD^EF`**

C. **`+-+AB*CD^EF`**

D. **`++-AB*CD^EF`**

E. None of these

QUESTION 37

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

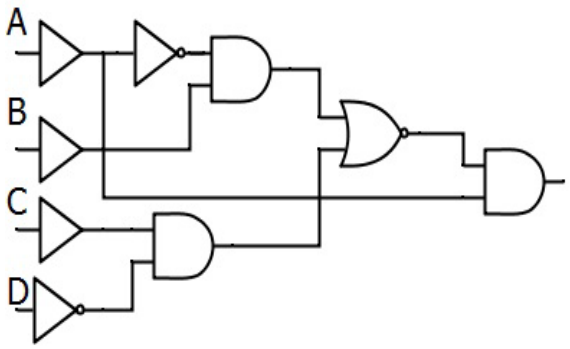
A. $\overline{(\overline{A}+B)*(C+\overline{D})}+A$

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

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

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

E. None of these



QUESTION 38

Find f(-10) according to the recursive function definition shown below.

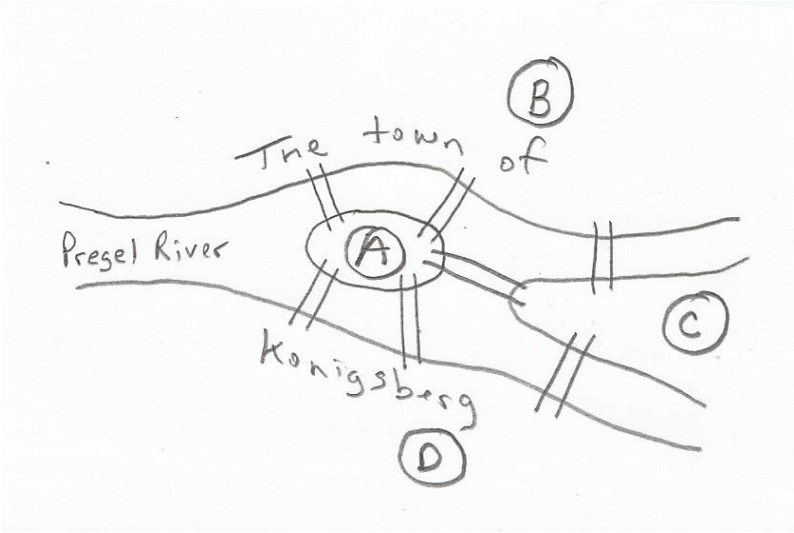
f(-10) =

```
f(x)=     f(x+5)+f(x+8)     when x<=0
f(x)=     2                 if x>0 and even
f(x)=     -1                if x>0 and odd
```

A. 0 B. 1 C. 2 D. 3 E. 4

QUESTION 39

A well-known graph problem studied by Leonard Euler had to do with the town of Konigsberg, in which there was an island at the point where the river Pregel forked. There were seven bridges connecting the island with the two banks of the river and the land between the forks, as shown in the picture. His question was whether or not there was a way to cross the seven bridges in a continuous walk through town without recrossing any of them. For example, he might start a path starting on the B side of town, cross over one of the two bridges to the island (A) and then on across to the south bank (D), and then across to the land area between the forks of the river (C). This path would be indicated by the letter sequence BADC. In this case study of the **Seven Bridges of Konigsberg**, a complete **Euler path**, or **Euler tour,** would be indicated by 8 letters. There could be any number of combinations.



Write the **sequence of eight letters** that represent your version of the Euler path for the seven bridges of Konigsberg. If you determine that it is not possible, write the answer "NOT POSSIBLE".

_ _ _ _ _ _ _ _

QUESTION 40

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

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

Computer Science Answer Sheet

Conference _____

Code Number _____

1. _____	11. _____	21. _____	31. _____
2. _____	12. _____	22. _____	32. _____
3. _____	13. _____	23. _____	33. _____
4. _____	14. _____	24. _____	34. _____
5. _____	15. _____	25. _____	35. _____
6. _____	16. _____	26. _____	36. _____
7. _____	17. _____	27. _____	37. _____
8. _____	18. _____	28. _____	38. _____
9. _____	19. _____	29. _____	39. _____
10. _____	20. _____	30. _____	40. _____

FOR GRADING USE ONLY

Number Correct _____ **x 6 =** _____

Number Incorrect _____ **x 2 =** _____

Grader 1 Initial _____

Grader 2 Initial _____

Grader 3 Initial _____

Subtract Line 2 above from Line 1.

SCORE

_____

Computer Science Answer Key
UIL Invitational B 2015

1)	D	11)	E	21)	C	31)	D
2)	E	12)	C	22)	B	32)	B
3)	B	13)	C	23)	C	33)	C
4)	E	14)	B	24)	D	34)	B
5)	B	15)	B	25)	C	35)	D
6)	A	16)	D	26)	A	36)	C
7)	A	17)	A	27)	B	37)	C
8)	A	18)	C	28)	D	38)	B
9)	D	19)	C	29)	C	39)	NOT POSSIBLE
10)	A	20)	A	30)	C	40)	$A * (\overline{B} + \overline{C})$

Note: Since AND and OR 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:

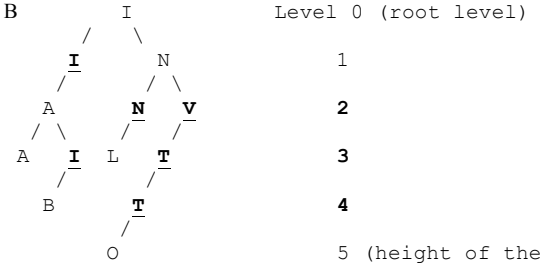
1. D $10111000_2 + AB_{16} = 184_{10} + 171_{10} = 355_{10} = 543_8 = 163_{16} = 101100011_2$
2. E 23 / 9 * 1 % 2 = 2 * 1 % 2 = 2 % 2 = 0
3. B Since a field width of 4 is not enough to contain 49.20, the compiler just takes what it needs and left justifies the output.
4. E The "IE" pattern finds the only "IE" in the word and replaces it with a single "O".
5. B Since p is true and q is false, p and q is false, which makes the NOT of P and Q true.
6. A This a lesser known Pythagorean triple...8, 15, 17. The Math.hypot method returns the hypotenuse for sides of 8 and 15, which when applied to the Pythagorean theorem becomes 64 + 225, or 289, whose square root is 17.
7. A 8 * 1.4 = 11.2, which truncates to 11 in the autocast provided by *=
8. A 14%3 equals 2, which results in a y value of 2 in the switch statement, and an output value of 14+2, or 16.
9. D The trace values of x are 4, 8, 9, 18, 19, 38, 39, 78, and 79, with the output value being 79.
10. A The sequence of the first loop is as follows: position 6 (5+1) of list2 gets the value ++a (2), 4 gets 3, 2 gets 4, 3 gets 5, 5 gets 6, and 1 gets 7. Positions 0 and 7 of list2 remain the default values 0. The second loop simply outputs the values of list2.
11. E For output purposes only, any of these statements will work with this data.
12. C The value sequence for d and x is: 100.0 0, 150.0 1, 225.0 2, 337.5 3, 506.25 4, 759.375 5, 1139.0625 6.
13. C The *expr++ expr--* (postfix) operators are on line 1 of the chart, followed by **|** on line 10, and **||** on line 12.
14. B Half of the values are negative (-128 to -1), and the other half non-negative (0-127).
15. B This toArray method of the ArrayList class returns an Object array, therefore the word Object must fill the <statement> in the code.
16. D

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

17. A 65 - 97 is equal to -32, which when tripled equals -96.
18. C The contents of the matrix after the assignment loop are:

```
0 0 45 30 15
0 36 21 6 0
0 12 0 42 27
0 48 33 18 3
0 24 9 0 39
```

19. C 118 in 8-bit binary is 01110110. Applying the rule mentioned in the problem gives you 10001010, which is the binary equivalent of -112.
20. A The worst case running time for a quick sort is O(N^2), which occurs when the list is already sorted, or when only a few elements are not in sorted order.
21. C This quick sort arranges a list of numbers in **descending** order due to the given comparison operators in statements 3 and 4. The output is the choice that shows all of the numbers in order from greatest to least.
22. B The comparison operators in both statements 3 and 4 need to be reversed for the quick sort to work in reverse order than is indicated here.
23. C The output string is "aeioueiouiououu", which contains four letter 'o's.
24. D The tangent of a 45 degree angle is 1.0 (opposite over adjacent)
25. C The value 10 in 8-bit binary is 00001010, which when made negative becomes 11110110 in twos complement form. When right shifted twice, it becomes 11111101, which is the decimal value -3.
26. A A precondition of this binary search algorithm is that the list is in ascending sorted order.
27. B The sorted list is -7, -5, -3, 0, 1, 3, 4, 5, 8, 9. Position 7 contains the 5, and -6 is not in the list, which is indicated by -1.
28. D The pattern ".*" matches zero or more of any character, ".+" matches one or more of any character, and the "?." matches one character, once or not at all.
29. C The recursive call value sequence for each initial call is: 6, 3, 0, result 2+2+1 = 5, 10, 7, 4, 1, -2, result 2+2+2+2+3 = 11, -5, result 3
30. C Since all alpha characters are above 60 in value, and all numeric and space characters are below, "A" is printed for the letters, and "B" is printed for the digits and the space character with this ternary operator statement.
31. D The base ten values of 7, 34, and 63 convert to base 8 values of 7, 42 (8 goes into 34 4 times with 2 remainder), and 77 (8 goes into 63 7 times with 7 remainder).
32. B



Level 0 (root level)

1

2

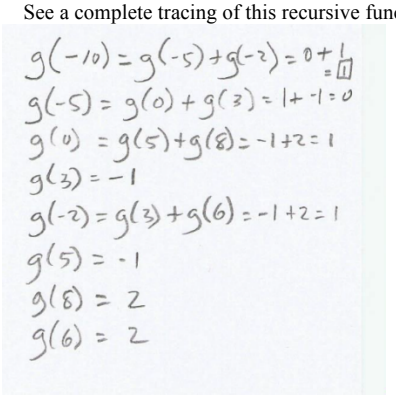
3

4

5 (height of the tree)

There are 6 parent nodes with only one child, shown in bold and underlined, and tree height is 4 (root node is at level zero, and furthest leaf is at level 4). In this binary search tree exercise, duplicates are allowed and are inserted to the left of matching elements. The first node is at level zero, therefore the resulting tree, as you can see in the key above, has a height of 4, with 5 leaves.

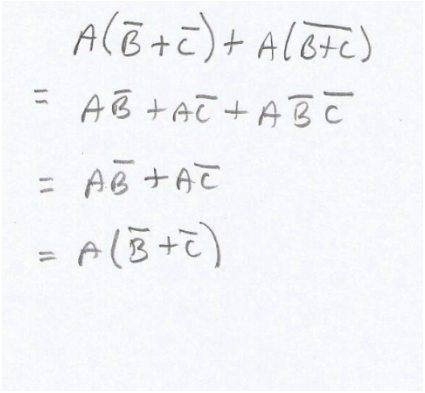
33. C The sequence of the first four statments is: call to default constructor(A), call to toString method(F) which returns and outputs the value 5, call to setThing(D) which changes the instance field value to 6, and a final call to the toString method(F), which returns and outputs the value 6.
34. B The sequence of the next two statments is: call to one parameter constructor(H) which assigns 3 to the ThingTwo instance field, but first automatically calls the ThingOne default constructor(A) which assigns 5 to the ThingOne instance field, followed by the call to the ThingTwo getThing method(I), which returns and outputs the instance field value of the ThingTwo object(3).
35. D The sequence of the last two statements is: call to reduceThing(K) which reduces the ThingTwo instance field value to 2, then call to the ThingTwo toString method(L), which in turn calls the ThingOne toString method(F), and returns concatenated values of both instance fields, 5 and 2.
36. C The first rule is that the operands stay in the same order, A B C D E F. Then follow order of operations. The exponent goes in front of the E and F, and the * before CD. The first + goes in front of the AB, then the minus before that, then finally the last + in front of everything.
37. C The signals NOT A and B go into an AND gate, which feeds into a NOT OR gate, which receives an AND signal from C and NOT D. The NOT OR result goes into an AND gate which also receives the A signal.
38. B See a complete tracing of this recursive function call shown above.



39. No matter where you start in this graph, it is **not possible** to cross all seven bridges without recrossing any of them. However, adding an 8th bridge, perhaps another between C and D, would enable an Euler path such as this: BADABCDCA. This problem helped give birth to the field of graph theory the development of which Euler is given much credit.

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

Explanation: In the diagram below, the distributive property is used to simplify the original expression. Absorption is used by the second term to eliminate the third term, and then the A is factored out, resulting in the final expression, which contains two operators, AND and OR, with NOTs over both the B and C terms.



University Interscholastic League
Computer Science Competition

Number 151 (District 1 - 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 $202_{10} + 10000101_2$?

A. 335_{10} B. 517_8 C. $14F_{16}$ D. 101001011_2 E. All are

QUESTION 2

What is output by the code segment to the right?

A. 4 B. 4.6 C. 18.2 D. 23.8 E. 24

```
out.println(17 + 7 - 1 / 5);
```

QUESTION 3

What is output by the code segment to the right?

A. A60 B. 6560 C. A1050 D. 651050 E. 125

```
out.println("A" + 10 + 50);
```

QUESTION 4

What is output by the code segment to the right?

A. 9 B. 10 C. 11 D. 18 E. 19

```
String s = "abcabcabcabcabcabcabc";
out.println(s.lastIndexOf("abc",10));
```

QUESTION 5

What is output by the code segment to the right?

A. false B. true

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

QUESTION 6

What is output by the code segment to the right?

A. 4 B. 4.0 C. 8 D. 8.0 E. 16

```
int x = 64;
out.println(Math.cbrt(x));
```

QUESTION 7

What is output by the code segment to the right?

A. -2.5 B. 2.5 C. 15.7 D. -15.7
E. There is no output due to an error.

```
long j = -55;
double d = -3.5;
out.println(j%d);
```

QUESTION 8

For which of these input values will the output of the code segment to the right be "yes"?

I 9
II 10
III 11
IV 19
V 21

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

```
int x = <input value>;
if(!(x>20||x<10))
 out.println("yes");
else
  out.println("no");
```

QUESTION 9

How many stars will be output by the code segment to the right?

A. 3 B. 4 C. 5 D. 6 E. 7

```
int x = 4;
String s = "";
do{
 s+="*";
 x+=2;
}while(x%7!=0);
out.println(s);
```

QUESTION 10

What is output by the code segment to the right?

A. 10 B. 106 C. 138 D. 202 E. 250

```
char[]list=new char[5];
list[1]='1';
list[2]='2';
list[3]=3;
list[4]=4;
out.println(list[0]+list[1]+list[2]+l
ist[3]+list[4]);
```

QUESTION 11

In the code segment to the right, which statement below must be placed in **<code block>** in order for this code segment to work properly?

A. `import java.io.*;`
B. `import java.util.*;`
C. `import static java.lang.System.*;`
D. `import static java.lang.Math.*;`
E. More than one of these.

```
<code block>
public class test{
 public static void main
                (String [] args){
   Scanner kb = new
        Scanner(System.in);
 }
}
```

QUESTION 12

What is output by the code segment to the right?

A. 10 55 B. 10 45 C. 11 45
D. 11 55 E. None of these

```
int x = 100;
int y = 1;
for(;y<10;y++)
 x-=y;
out.println(y+" "+x);
```

QUESTION 13

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

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

I. ^
II. ||
III. ? :

QUESTION 14

What is output by the code segment to the right?

A. 8 B. 16 C. 32 D. 64 E. None of these

```
out.println(Character.SIZE);
```

QUESTION 15

What is output by the code segment to the right?

A. `false false` B. `false true` C. `true false`
D. `true true` E. There is no output due to an error.

```
int [] pList = {10,20,30,40,50};
ArrayList<Integer> aList = new
  ArrayList<Integer>();
for(int a:pList)
  aList.add(a/10);
out.print(pList.contains(30)+ " ");
out.println(aList.contains(30));
```

QUESTION 16

How many ordered pairs make this boolean expression false?

A. 0 B. 1 C. 2 D. 3 E. 4

$$\overline{\bar{A} + B}$$

QUESTION 17

What is output by the code to the right?

A. 7 B. 7.0 C. 17 D. 17.0

E. There is no output due to an error.

```
long j = 20;
int k = -15;
double p = 5;
out.println(j+k/p);
```

QUESTION 18

What is output by the code segment to the right?

A. 3 B. 4 C. 5 D. 6 E. 7

```
int [][] grid = new int[5][4];
for(int r=1;r<grid.length;r++)
 for(int c=0;c<grid[r].length;c++)
  grid[r][c] = r+c;
out.println(grid[3][2]);
```

QUESTION 19

Which of the following choices represents the decimal equivalent of the two's complement binary value 11001010?

A. -51 B. -52 C. -53 D. -54 E. -55

QUESTION 20

What is the output at the end of the third iteration in the method execution called by the client code to the right?

A. 8 4 2 7 6 B. 2 4 8 7 6 C. 8 7 4 2 6
D. 2 4 7 8 6 E. 8 7 6 4 2

QUESTION 21

What algorithm is represented by the method `mystD1`?

A. Insertion sort B. Selection sort C. Bubble sort
D. Merge sort E. Quick sort

QUESTION 22

In which line of this method must a change be made so that the sorting order is reversed?

A. Line 2 B. Line 4 C. Line 5 D. Line 6 E. Line 8

QUESTION 23

What is the least restrictive order of magnitude for the average case in the sort shown in the code to the right?

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

```
1 public static void mystD1(int[] list){
2  for (int j = 1; j < list.length; j++){
3   int temp = list[j];
4   int i = j;
5   while (i > 0 && temp > list[i - 1]){
6     list[i] = list[i - 1];
7     i--;
     }
8   list[i] = temp;
9   for(int x:list)
10     out.print(x+" ");
11   out.println();
  }

<client code>
int [] list = {8,2,4,7,6};
mystD1(list);
```

QUESTION 24

What is the output of the code segment to the right?

A.
```
43210
6543210
10
3210
543210
```
B.
```
3210
543210
0
210
43210
```
C.
```
EEEEE
GGGGGGG
BB
DDDD
FFFFFF
```
D.
```
EEEE
GGGGGG
B
DDD
FFFFF
```
E. There is no output due to an error.

```
String s = "EGBDF";
char [] list = s.toCharArray();
for(char a:list){
 int x = a-64;
 while(x-->0)
    out.print(a);
 out.println();
}
```

QUESTION 25

What is output by the code segment to the right?

A. 30.0 B. 45.0 C. 60.0 D. 90.0 E. None of these

```
double val = 0.5;
out.printf("%.1f\n",
   Math.toDegrees(Math.acos(val)));
```

QUESTION 26

What is output by the code segment to the right?

A. `1.23.456.095.4`
B. `1.2 3.45 6.09 5.4`
C. `[1.2 3.45 6.09 5.4]`
D. `[1.2, 3.45, 6.09, 5.4]`
E. There is no output due to an error

```
double [] list = {1.2,3.45,6.09,5.4};
out.println(Arrays.toString(list));
```

QUESTION 27

Which choice below represents the two terms listed in order to correctly replace <thing1> and <thing2> in the method definition to the right?

A. `int` and `return` B. `return` and `int`

C. `public` and `return` D. `public` and `int`

E. `return` and `public`

```
static <thing1> mystD1 (int x){
 if(x%4==0)
   <thing2> x*5;
 <thing2> x;
}
```

QUESTION 28

Assuming <thing1> and <thing2> have been correctly replaced, what is the output of the client code to the right?

A. `1 0` B. `1 40` C. `45 0` D. `9 8` E. `9 40`

```
<client code>
out.print(mystD1(9)+ " ");
out.println(mystD1(8));
```

QUESTION 29

In the Dwelling class definition to the right, how many instance fields are there?

A. 1 B. 2 C. 3 D. 4 E. 5

QUESTION 30

Using the Dwelling class definition to the right, with the block comment symbols (**`/* */`**) in place as shown, what is the output of the client code below?

```
Dwelling d = new Dwelling();
out.println(d);
```

A. `null 0 false`
B. `Dwelling@150bd4d`
C. `null 0 rooms not sturdy`
D. `: 0 rooms: not sturdy`
E. `null: 0 rooms: not sturdy`

QUESTION 31

If the block comment symbols are removed in the Dwelling class definition to the right, what is the output of the client code below?

```
Dwelling d = new Dwelling();
out.println(d);
d = new Dwelling("tent",2,false);
out.println(d);
```

A.
```
house: 3 rooms: sturdy
tent: 2 rooms: not sturdy
```
B.
```
Dwelling@150bd4d
Dwelling@1bc4459
```
C.
```
house: 3: true
tent: 2: false
```
D.
```
house: 3 rooms: true
tent: 2 rooms: false
```
E.
There is no output due to an error.

```
class Dwelling{
 private String type;
 private int numRooms;
 private boolean sturdy;
/*
 public Dwelling(){
   type = "house";
   numRooms = 3;
   sturdy = true;
 }
 public Dwelling(String ty, int nr,
                 boolean st){
   type = ty;
   numRooms = nr;
   sturdy = st;
 }
*/
 public String toString(){
   return type+": "+numRooms
     +" rooms: " +
     (sturdy?"sturdy":"not sturdy");
 }}
```

QUESTION 32

What is output by the code to the right?

A. 2 B. 2.5 C. 22 D. 60

E. There is no output due to an error

```
out.println(Integer.toString(12,5));
```

QUESTION 33

Using the generic stack pseudocode on the right, what is the sum of all popped items after the push and pop sequence is complete?

A. 14 B. 17 C. 18 D. 22 E. 25

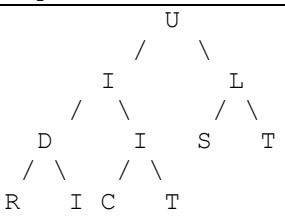
```
Push 3
Push 9
Push 6
Push 4
Pop x
Pop x
Push 5
Pop x
Push 3
Pop x
```

QUESTION 34

What is the preorder traversal of the binary tree shown to the right?

A. UIDRIICTLST B. RDIICITUSLT C. RIDCTIISTLU

D. UILDISTRICT E. ULTSIITCDIR

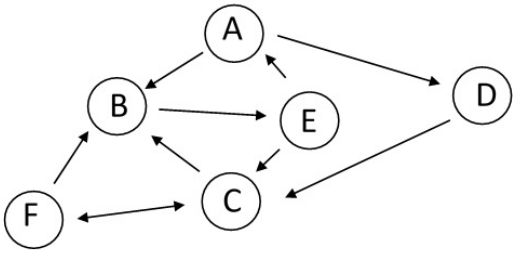


QUESTION 35

Which of the following is **NOT** a simple path in the graph shown to the right?

I. ABEC
II. EABEC
III. DCBEA
IV. DCBF

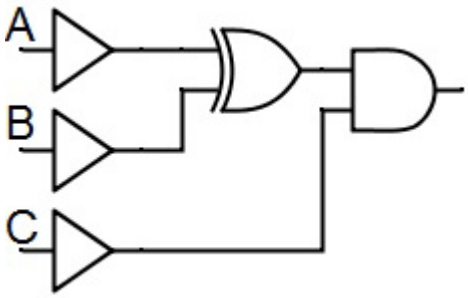
A. I only B. II only C. IV only D. I and III only E. II and IV only



QUESTION 36

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

A. $\overline{A \oplus B} * C$
B. $A \oplus B * C$
C. $(A + B) * C$
D. $(A \oplus B) * C$
E. None of these



QUESTION 37

Which of the expressions below is the postfix equivalent to the expression shown?

A. **- + I / N ^ T E L**
B. **I N T E ^ / - L +**
C. **I N T E ^ / L + -**
D. **- I + / N ^ T E L**
E. None of these

I + N / T ^ E - L

QUESTION 38

Which of the following values is NOT a possible outcome of the code shown to the right?

A. 33 B. 35 C. 36 D. 37 E. 38

```
Random r = new Random();
out.print(r.nextInt(5)+33);
```

QUESTION 39

Free Response Question:
Simplify the Boolean Algebra expression shown below as much as possible.

$$\overline{A * B + \overline{A + B}}$$

QUESTION 40

Free Response Question:

Find f(5) according to the recursive function definition shown below.

f(5) =

```
           f(x-3)+1     when x>0
f(x)=      3            when x=0
           2            when x<0
```

NO TEST MATERIAL ON THIS PAGE

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 Sheet

Conference _____

Code Number _____

1. _____	11. _____	21. _____	31. _____
2. _____	12. _____	22. _____	32. _____
3. _____	13. _____	23. _____	33. _____
4. _____	14. _____	24. _____	34. _____
5. _____	15. _____	25. _____	35. _____
6. _____	16. _____	26. _____	36. _____
7. _____	17. _____	27. _____	37. _____
8. _____	18. _____	28. _____	38. _____
9. _____	19. _____	29. _____	39. _____
10. _____	20. _____	30. _____	40. _____

FOR GRADING USE ONLY

Number Correct _____ **x 6 =** _____

Number Incorrect _____ **x 2 =** _____

Grader 1 Initial _____

Grader 2 Initial _____

Grader 3 Initial _____

Subtract Line 2 above from Line 1.

SCORE

_____

Computer Science Answer Key
UIL District 1 2015

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

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. 202_{10} + 10000101_{2} = 202_{10} + 133_{10} = 335_{10} = 517_{8} = $14F_{16}$ = 101001111_{2}
2. `17 + 7 - 1 / 5 = 17 + 7 - 0 = 17 + 7 = 24`
3. `"A" + 10 + 50 ==> "A10" + 50 ==> "A1050"`
4. Starting backwards from position 10 in the string, the last index of "abc" is 9.
5. Since p is true, NOT p is false, which makes the AND result false, regardless of q's value.
6. The cube root of 64 is 4.0.
7. The mod operator always results in a sign the same as the first operand, regardless of the sign of the second operand. Therefore, -55 mod -3.5 is the same as -55 mod 3.5, which is -2.5.
8. The values 10, 11, and 19 all make the OR conditional statement false, which when NOTed, becomes true, thus resulting in the "yes" output.
9. One star will be concatenated to the string s for each of the x values 4, 6, 8, 10, and 12. At 14, the while statement becomes false since 14 % 7 is equal to zero, resulting in an output of 5 stars.
10. The five values in the array are 0, 49, 50, 3, and 4, which sum to 106.
11. The Scanner class "lives" in the java.util package, which is the only one essential to this code for keyboard input. The java.io package must be included when using the File class for file input.
12. The sequence of values for y and x is: 1 100, 1 99, 2 97, 3 94, 4 90, 5 85, 6 79, 7 72, 8 64, 9 55, 10 55
13. The bitwise XOR operator (^) is on line 9 of the chart, followed by the logical OR operator (**||**) on line 12, and the ternary operators (**?:**) on line 13.
14. The Character data type uses 16 bits of storage.
15. The contains method does not work with the primitive array pList, thus the error. If it did, the answer would be **true false**, since pList does contain the value 30, but aList does not since the value 30 was divided by 10 first, then added to aList.
16. $A \quad B \quad \overline{A} \quad \overline{A}+B \quad \overline{\overline{A}+B}$

0	0	1	1	**0**
0	1	1	1	**0**
1	0	0	0	**1**
1	1	0	1	**0**

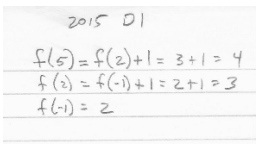
The truth table above shows three false results in the final column.

$\overline{\overline{A}+B}$ simplifies to $A * \overline{B}$ using DeMorgan's law and the Double Negative Rule, showing (1,0) as the only true result.

17. `20 + -15 / 5 ==> 20 + -3.0 ==> 17.0`
18. The value of each matrix element is the sum of the row and column index, which in the case of grid[3][2] is 3+2, which is 5.
19. Using the two's complement short-cut conversion process (see either 2015 Invitational Test for a complete explanation), 11001010 converts back to 00110110, which is the value 54, hence the original bit string is -54.
20. This is the classic insertion sort (descending order), where one by one, from the second element to the end of the list, the best place is found for each element. In the first iteration, nothing changed, since the 2 was already in the best slot at that time. In the second iteration, the 4 was swapped with the 2. On the third iteration, the 7 was moved towards the front two slots, moving the 4 and 2 to the right, resulting in the order 8 7 4 2 6.
21. This is the insertion sort, as described above.
22. In line 5, **`temp > list[i - 1])`** (descending order) must be changed to **`temp < list[i - 1],`** which will make sort in ascending order.
23. The order of magnitude for the average and worst cases of the insertion sort is O(N^2), with O(N) for the best case.
24. This process outputs each letter of the original string the number of times corresponding to its position in the alphabet, i.e., the E is output five times, the G seven times, etc.
25. The angle whose cosine is 0.5 measures 60 degrees
26. The Arrays toString method outputs any char, double, int, long, or Object array in this form: `[1.2, 3.45, 6.09, 5.4]`
27. Since this is a method that is called in an output statement, it is a return method, which means <thing1> is the type data being returned, `int`, and <thing2> is the actual return statement, requiring the term `return`.
28. Since 9 mod 4 is not zero, 9 is the value returned. For 8, whose mod 4 value is 0, 8 * 5, or 40 is returned.
29. The three instance fields are: `private String type;private int numRooms; private boolean sturdy;`
30. With block comments hiding both constructors, the compiler assigns default values to the three instance fields, which are then output using the toString method in the format shown, with null, zero, and false as the default values.
31. With both constructors uncommented, the provided values are shown in the output from the toString method.
32. The base 5 equivalent of 12 is 22, which essentially means 5 goes into 12 2 times, with 2 remainder.
33. The four popped values in sequence are 4, 6, 5, and 3, which sum to 18.
34. Preorder traversal starts at the root of the tree (top node), and "touches left" each node along all of the branches of the tree from left to right.
35. EABEC is not a simple path because it revisits a node, and DCBF is not a path at all since there is no path from B to F, only from F to B.
36. Since XOR has a lower precedence than AND, parentheses must surround the XOR expression since it comes first in the diagram.
37. **`I N T E ^ / + L -`** is the correct postfix equivalent for this expression.
38. The 5 possible values generated by this code range from 33 through 37, therefore 38 is not possible.
39. $A \oplus B$

Rationale:

$\overline{A * B + \overline{A} + \overline{B}}$ becomes $(\overline{A} + \overline{B}) * (A + B)$ - DeMorgan's law on the * and the +, and then double negative, which then becomes $\overline{A} * A + \overline{A} * B + \overline{B} * A + \overline{B} * B$ using FOIL. The complement law further simplifies it to $\overline{A} * B + A * \overline{B}$, which is the definition for $A \oplus B$

40. 

University Interscholastic League
Computer Science Competition

Number 152 (District 2 - 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 $11010001_2 - 42_{10}$?

A. 167_{10} B. 247_8 C. $A7_{16}$ D. 10100111_2 E. All are equivalent

QUESTION 2

What is output by the code segment to the right?

A. 0 B. 0.55 C. 4 D. 24

E. undefined - mod by zero error

```
out.println(24 * 1 % 2 / 3);
```

QUESTION 3

What is the output of the code segment to the right?

A. 1000 B. 952 C. 9502 D. 948

E. There is no output due to an error

```
out.println(1000 - 50 + "2");
```

QUESTION 4

What is output by the code segment to the right?

A. true B. false C. There is no output due to an error

```
String s = "jazzyjazzygirl";
String t = "jazmine";
out.println(s.startsWith
            (t.substring(1,3),6));
```

QUESTION 5

What is output by the code segment to the right?

A. false B. true

```
p = false;
q = false;
out.println(!(p&&q));
```

QUESTION 6

What is output by the code segment to the right?

A. 47 B. -47.9 C. 47.9 D. 48 E. 47.0

```
double d = 47.9;
out.println(Math.abs(d));
```

QUESTION 7

What is output by the code segment to the right?

A. 3 B. 3.0 C. 3f D. 3.0f

E. There is no output due to an error.

```
float f = 1.0;
int x = 2;
out.println(f+x);
```

QUESTION 8

For which of these input values will the output of the code segment to the right be "blue"?

I -30
II -40
III -14
IV -16
V -39

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

D. I, IV, and V only E. III and V only

```
int x = <input value>;
if(!(x<-15&&x>-40))
 out.println("red");
else
  out.println("blue");
```

QUESTION 9

What is output by the code segment to the right?

A. `UILComputerSci ence2015`

B. `UILComputerSc ence2015`

C. `UILComputerSc ience2015`

D. Nothing is output

E. There is no output due to an error

```
String s = "UILComputerScience2015";
String t = "";
while(!t.contains("i"))
{
 t+=s.charAt(0);
 s=s.substring(1);
}
out.println(t+" "+s);
```

QUESTION 10

What is output by the code segment to the right?

A. `12345`

B. `1.02.03.04.05.0`

C. `1.0000002.0000003.0000004.0000005.000000`

D. `15`

E. There is no output due to an error

```
double [] list = {1,2,3,4,5};
for(double d:list)
  out.print(d);
```

QUESTION 11

In the code segment to the right, which statement below must be placed in **<code block>** in order for this code to work properly?

A. `import java.io.*;`

B. `import java.util.*;`

C. `import static java.lang.System.*;`

D. `import static java.lang.Math.*;`

E. More than one of these.

```
<code block>
public class test{
 public static void main
           (String [] args)
               throws IOException{
   Scanner f = new Scanner
           (new File("stuff.dat"));
 }
}
```

QUESTION 12

What is output by the code segment to the right?

A. `31 28 6` B. `31 28 7` C. `31 29 6` D. `30 30 7`

E. None of these

```
int x = 10;
int y = 50;
int z = 1;
for(;x<=y;z++){
 x+=z;
 y-=z;
}
out.println(x+" "+y+" "+z);
```

QUESTION 13

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

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

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

I. `&`
II. `|`
III. `+ -`

QUESTION 14

What is output by the code segment to the right?

A. `8` B. `16` C. `32` D. `64` E. None of these

```
out.println(Long.SIZE);
```

QUESTION 15

What is output by the code segment to the right?

A. `14 9` B. `9 14` C. `15 10` D. `10 15`

E. There is no output due to an error.

```
int [] pList =
 {1,2,3,1,2,3,1,2,3,1,2,3,1,2,3,2,3};
ArrayList<Integer> aList = new
    ArrayList<Integer>();
for(int a:pList)
    aList.add(a);
out.print(aList.lastIndexOf(2)+" ");
Collections.sort(aList);
out.println(aList.lastIndexOf(2));
```

QUESTION 16

What is the output after the second iteration during the execution of the code segment to the right using the client code shown below?

```
int [] list = {8,2,4,7,6};
mystD2(list);
```

A. 2 4 8 7 6 B. 2 4 6 7 8 C. 8 2 4 7 6

D. 8 7 4 2 6 E. 2 8 4 7 6

```
1 public static void mystD2(int[] list)
2 {
3   for (int x = 0; x < list.length - 1;
              x++){
4    int b = x;
5    for (int y = x+1; y < list.length;
              y++)
6      if (list[y] < list[b])
7        b = y;
8    int a = list[x];
9    list[x]=list[b];
10   list[b]=a;
11   for(int i:list) out.print(i+" ");
12      out.println();
   }
}
```

QUESTION 17

What algorithm is represented by the method `mystD2`?

A. Insertion sort B. Selection sort C. Bubble sort

D. Merge sort E. Quick sort

QUESTION 18

In which line of `mystD2` must a change be made so that the sorting order is reversed?

A. Line 3 B. Line 5 C. Line 6

D. Line 8 E. Line 10

QUESTION 19

What is the least restrictive order of magnitude for the average case in the sort shown in the code to the right?

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

QUESTION 20

Which of the following choices represents the decimal equivalent of the two's complement binary value 11011100?

A. -36 B. -37 C. -38 D. -39 E. -40

QUESTION 21

How many ordered pairs make this boolean expression false?

A. 0 B. 1 C. 2 D. 3 E. 4

$$\overline{\bar{A} * \bar{B}}$$

QUESTION 22

What is the postorder traversal of the binary tree shown to the right?

A. COPSMCI B. PSOCIMC C. POSCCMI

D. COMPSCI E. ICMSPOC

QUESTION 23

What is output by the code segment to the right?

A. 4 B. 4.0 C. -4 D. -4.0

E. There is no output due to an error.

```
double d = 9.5;
int i = 14;
int j = 0;
j += d -= i;
out.printf("%d",j);
```

QUESTION 24

Using the generic queue pseudocode on the right, what is the sum of all popped items after the push (enqueue) and pop (dequeue) sequence is complete?

A. 14 B. 17 C. 18 D. 22 E. 25

```
Push 3
Push 9
Push 6
Push 4
Pop x
Pop x
Push 5
Pop x
Push 2
Pop x
```

QUESTION 25

What is the output of the code segment to the right?

A. 54 10 B. 54 12 C. 54 13 D. 66 10 E. 66 13

```
int s=0,t=0;
while(s<50){
 t=10;
 do{
    s+=t++;
 }while(t<=12);
}
out.println(s+" "+t);
```

QUESTION 26

In the Weather class definition to the right, how many constructors are there?

A. 1 B. 2 C. 3 D. 4 E. 5

QUESTION 27

Using the Weather class definition to the right, with the block comment symbols in place as shown, what is the output of the client code below?

```
Weather d = new Weather();
out.println(d);
```

A. rain: 65 degrees: wet

B. Weather@4a5ab2

C. rain: 65: dry

D. :0 : false

E. null: 0: false

QUESTION 28

If the block comment symbols are removed in the Weather class definition to the right, what is the output of the client code below?

```
Weather d = new Weather("fair",74,false);
out.println(d);
```

A. Weather@4a5ab2
B. fair: 74: dry
C. fair: 74 degrees: dry
D. fair: 74 degrees: wet
E. There is no output due to an error

```
class Weather{
 private String type;
 private int temperature;
 private boolean wet;

 public Weather(){
    type = "rain";
    temperature = 65;
    wet = true;
 }
 public Weather(String ty, int te,
                boolean wt){
    type = ty;
    temperature = te;
    wet = wt;
 }
/*
 public String toString(){
    return type+": "+temperature+"
           degrees: " +
                (wet?"wet":"dry");
*/
 }
}
```

QUESTION 29

Which of the following values is **NOT** a possible outcome of the code shown to the right?

A. 25 B. 27 C. 29 D. 31 E. 33

```
Random r = new Random();
out.print(r.nextInt(8)+26);
```

QUESTION 30

What is output by the code segment below?

```
char[][]list={"freshman".toCharArray(),"sophomore".toCharArray(),
              "junior".toCharArray(),"senior".toCharArray()};
int x = 0;
for(char[]a:list)
 for(char b:a)
  x+=b=='o'?1:0;
out.println(x);
```

A. 1 B. 2 C. 3 D. 4 E. 5

QUESTION 31

Which of the expressions below is equivalent to the expression shown to the right?

A. **M * A ^ R * T / I - N**
B. **M * A ^ R * T - I / N**
C. **M ^ A * R * T / I - N**
D. **M * A * R ^ T / I - N**
E. None of these

- / * * M ^ A R T I N

QUESTION 32

What is output by the code segment to the right?

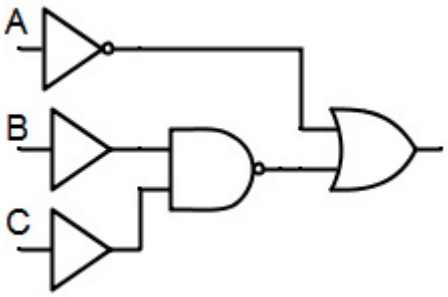
A. 30.0 B. 45.0 C. 60.0 D. 90.0 E. None of these

```
double val = 0.5;
out.printf("%.1f\n",Math.toDegrees
            (Math.asin(val)));
```

QUESTION 33

Which of the following logical statements represents the diagram on the right?

A. $A + B * C$
B. $\bar{A} * \overline{B + C}$
C. $\bar{A} + B * C$
D. $A + \overline{B * C}$
E. None of these



QUESTION 34

Which of the choices below correctly fills <statement> in the code segment to the right in order to produce the output shown below?

[0, 0, 5, 5, 5, 5, 5, 0, 0, 0]

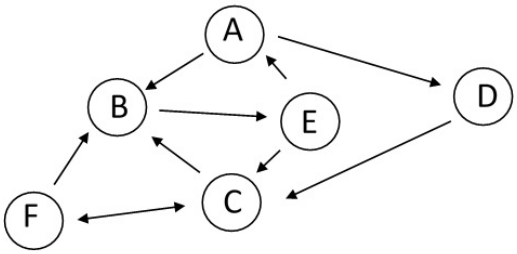
A. `Arrays.fill(list, 2, 6, 5);`
B. `Arrays.fill(list, 2, 7, 5);`
C. `Arrays.fill(list, 5, 7, 2);`
D. `Arrays.fill(list, 5, 6, 2);`
E. None of these

```
int [] list = new int [10];
<statement>
out.println(Arrays.toString(list));
```

QUESTION 35

In the graph shown to the right, how many different cycles are there?

A. 0 B. 2 C. 4 D. 6 E. more than 6



QUESTION 36

Which choice below represents the term listed to correctly replace `<term>` in the method to the right?

A. `public` B. `static` C. `void`
D. `int` E. `return`

QUESTION 37

Assuming `<term>` has been correctly replaced, what values are output by the client code to the right?

A. `10 50 16 16` B. `10 10 16 80`
C. `10 10 16 16` D. `50 50 80 80`
E. None of these

```
static <term> mystD2 (int x){
 int y=0;
 if(x%5==1)
    y = x*5;
 else
    y = x;
 out.println(x+" "+y);
}

<client code>
mystD2(10);
mystD2(16);
```

QUESTION 38

What is output by the code to the right?

A. `112` B. `88` C. `5` D. `5.5`
E. There is no output due to an error

```
out.println(Integer.toString(22,4));
```

QUESTION 39

Free Response Question: Simplify the Boolean Algebra expression shown below as much as possible.

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

QUESTION 40

Free Response Question: Find f(9) according to the recursive function definition shown below.

f(9) =

```
          f(x-3)+1    when x>0
f(x)=     f(x+2)-2    when x=0
          1           when x<0
```

NO TEST MATERIAL ON THIS PAGE

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 Sheet

Conference _____

Code Number _____

1. _____	11. _____	21. _____	31. _____
2. _____	12. _____	22. _____	32. _____
3. _____	13. _____	23. _____	33. _____
4. _____	14. _____	24. _____	34. _____
5. _____	15. _____	25. _____	35. _____
6. _____	16. _____	26. _____	36. _____
7. _____	17. _____	27. _____	37. _____
8. _____	18. _____	28. _____	38. _____
9. _____	19. _____	29. _____	39. _____
10. _____	20. _____	30. _____	40. _____

FOR GRADING USE ONLY

Number Correct _____ **x 6 =** _____

Number Incorrect _____ **x 2 =** _____

Grader 1 Initial _____

Grader 2 Initial _____

Grader 3 Initial _____

Subtract Line 2 above from Line 1.

SCORE

_____

Computer Science Answer Key
UIL District 2 2015

1)	E	11)	E	21)	B	31)	A
2)	A	12)	E	22)	B	32)	A
3)	C	13)	E	23)	C	33)	E
4)	A	14)	D	24)	D	34)	B
5)	B	15)	C	25)	E	35)	D
6)	C	16)	A	26)	B	36)	C
7)	E	17)	B	27)	B	37)	B
8)	D	18)	C	28)	C	38)	A
9)	A	19)	B	29)	A	39)	0 (zero)
10)	B	20)	A	30)	E	40)	3

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.

This page intentionally left blank

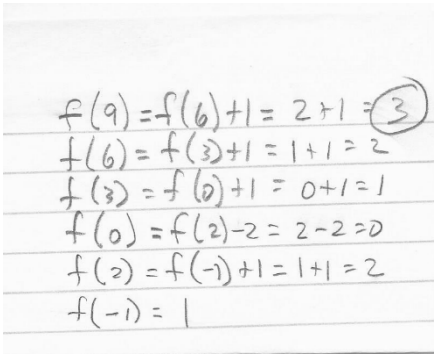
Explanations:

1. $11010001_2 - 42_{10} = 209_{10} - 42_{10} = 167_{10} = 247_8 = A7_{16} = 10100111_2$
2. Since all of these operators are of equal precedence, the evaluation simply goes from left to right:
   `24 * 1 % 2 / 3 = 24 % 2 / 3 = 0 / 3 = 0`
3. `1000 - 50 + "2" ==> 950 + "2" ==> "9502"`
4. Since the substring at 1,3 is "az", and the portion of the "jazzyjazzygirl" at position 6 to the end is "azzygirl", this statement outputs true.
5. The AND result with both p and q false, is false, therefore the NOT result of that is true.
6. The absolute value of 47.9 is 47.9.
7. To assign to a float variable, you must put an f after the value. The correct assignment is float f = 1.0f.
8. The values -30 and , -16, and -39 all make the AND conditional statement true, which when NOTed, becomes false, thus resulting in the "blue" output.
9. The loop builds up String t from the characters of String s, depleting String s one character at a time, until an "i" is encountered, at which time the loop ends. The "i" is added to t BEFORE the loop exits, resulting in the choice indicated.
10. Although the initial assignment contains integers, they are promoted to double status, and output with only one zero after the decimal point.
11. The Scanner class "lives" in the java.util package, and the File class is in the java.io package. Both must be imported for file input.
12. The sequence of values for x, y and z are: 10 50 1, 11 49 1, 13 47 2, 16 44 3, 20 40 4, 25 35 5, 31 29 6, and 31 29 7.
13. The **additive** operators (+−) are on line 4 of the chart, followed by the bitwise AND operator **(&)** on line 8, and bitwise OR (|) on line 10.
14. The Long data type uses 64 bits of storage.
15. The lastIndexOf method returns the position of the last instance of a value in the list, in this case, 15 before the sort, and 10 after the sort.
16. Since this is sorting in ascending order, in the first pass, the 2 is selected as the best value for position 0, and in the second pass, the 4 is moved into position 1, with the 8 moving towards the back each time. The order after the second pass is 2 4 8 7 6.
17. This is the selection sort.
18. In line 6, **`if (list[y] < list[b])`** must be changed to **`if (list[y] > list[b])`** to reverse the order of the sort.
19. The order of magnitude for all cases of the selection sort is O(N^2).
20. Using the two's complement short-cut conversion process (see either 2015 Invitational Test for a complete explanation), 11011100 converts back to 00100100, which is the value 36, hence the original bit string is -36.
21. A B $\overline{A}$ $\overline{B}$ $\overline{A} * \overline{B}$ $\overline{\overline{A} * \overline{B}}$

0	0	1	1	**1**	**0**
0	1	1	0	**0**	**1**
1	0	0	1	**0**	**1**
1	1	0	0	**0**	**1**

The truth table above shows only one false result in the final column.
$\overline{\overline{A} * \overline{B}}$ simplifies to $A + B$ using DeMorgan's law and the Double Negative Rule, showing true when either A is true or B is true, of which there are three pairs that work: (0,1), (1,0), and (1,1), with (0,0) producing the only false result.

22. Postorder traversal starts at the root of the tree (top node), and "touches right" each node along all of the branches of the tree from left to right.
23. 0 + (9.5 - 14) ==> 0 + (-4.5) ==> -4. Due to autocasting, there is no error, and the decimal portion is truncated in the final operation.
24. The four popped values in sequence are 3, 9, 6, and 4, which sum to 22.
25. The sequence of s and t values is: 0 0, 0 10, 10 11, 21 12, 33 13, 33 10 43 11, 54 12, 66 13.
26. The two constructors are the default (empty parameter list), and the one with three parameters.

27. Since the toString method is hidden by the block comment symbols, the default output is provided by the compiler, as shown in choice b.
28. Now that the toString method is active, the output is customized accordingly.
29. The 8 values generated by this code range from 26 through 33.
30. There are five instances of the letter 'o' in the matrix.
31. Since the exponent goes first, it goes between the A and R, then the two * between M and A, then between R and T, followed by the division and subtraction in the last two spots.
32. The angle whose sine is 0.5 measures 30 degrees
33. $\bar{A} + \overline{B * C}$ is the correct expression for this diagram.
34. The statement `Arrays.fill(list, 2, 7, 5);` fills positions 2 through 6 with the value 5.
35. A cycle is defined as a simple path (no revisited nodes in the middle) that ends where it started. ABEA is a cycle, which can also be named BEAB, or EABE. The other cycles are: ADCBEA, ADCFBEA, BECB, BECFB, and CFC.
36. Since this is a method called in a standalone statement, it is a void method, which means <term> is replaced by the word `void`.
37. Since 10 mod 5 is not equal to 1, y is assigned the value 10, producing an output of `10 10`. 16 mod 5 is equal to 1, therefore y is assigned the value 16 times 5, which is 80, with an output of `16 80`.
38. The base 4 equivalent of 22 is 112, which essentially means 4^2 (16) goes into 22 once, and then 4 goes into the remainder of 6 once, with a final remainder of 2.
39. $\overline{(A + \bar{B}) + \bar{A}}$ becomes $\bar{A} * B * A$ by DeMorgan's law on both +s, and the double negative law, which then becomes $0 * B$ because of the complement law, and finally just 0 because of the zero identity law
40. 

University Interscholastic League
Computer Science Competition

Number 153 (Region - 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 $74_{16} + 1101010_2$?

A. 11011110_2 B. 336_8 C. 220_{10} D. DE_{16} E. All are equivalent

QUESTION 2

What is the result of the expression shown?

A. 43 B. 91 C. 9 D. 25 E. 0

```
out.println(21 / 3 * 5 + 8);
```

QUESTION 3

In an output statement such as the one shown below, which choice below correctly labels each part of the statement?

`System.out.println("Hello");`

A. `System`-object, `out`-class, `println`-method, `"Hello"`-parameter
B. `System`-class, `out`-method, `println`-object, `"Hello"`-parameter
C. `System`-class, `out`-object, `println`-parameter, `"Hello"`-method
D. `System`-class, `out`-object, `println`-method, `"Hello"`-parameter
E. `System`-parameter, `out`-object, `println`-method, `"Hello"`-class

QUESTION 4

What is output by the code segment below?

```
String s = "Sara's seven sisters slept soundly in sand.";
out.println(s.replace("s","th"));
```

A. `Sara'th seven sisters slept soundly in sand.`
B. `Thara's theven thithterth thlept thoundly in thand.`
C. `Thara's seven sisters slept soundly in sand.`
D. `Sara'th theven thithterth thlept thoundly in thand.`
E. There is no output due to an error.

QUESTION 5

What is output by the code segment to the right?

A. `false` B. `true`

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

QUESTION 6

What is output by the code segment to the right?

A. 3 B. 3.0 C. 1.5 D. 2.0
E. There is no output due to an error.

```
int x = 1000;
out.println(Math.log10(x));
```

QUESTION 7

What is output by the code to the right?

A. 66.7 B. 66 C. 34.7 D. B
E. There is no output due to an error.

```
float f = 33.3f;
char a = 'd';
out.println(a - f);
```

QUESTION 8

What is output by the code to the right?

A. 6
B. 18
C. 60
D. 48
E. 160

```
int a = 0;
for(int x=0;x<10;x++){
      switch(x%5) {
            case 0: a++;break;
            case 1: a-=2;
            case 2: a+=5;break;
            case 3: a--;
            default: a=a+a;
      }
}
out.println(a);
```

QUESTION 9

What is output by the code to the right?

A. 10 B. 8 C. 6
D. There no output due to an infinite loop
E. There no output due to an error

```
int a = 100;
while(a>10) a/=1.5;
out.println(a);
```

QUESTION 10

Which of the original elements of the array shown to the right is no longer in the array after execution of the code?

A. 1 B. 2 C. 3 D. 4 E. 5

```
int [] list = {5,3,1,4,2};
list[3] = list[list[2]];
list[list[2]] = list[4];
```

QUESTION 11

Which of the statements below will NOT work properly when inserted into the code to the right?

I. `k.next().charAt(4);`
II. `k.nextChar(4);`
III. `k.nextLine().charAt(4);`

A. I only B. II only C. III only
D. I and III only E. All will work properly

```
String s = "grandbaby";
Scanner k = new Scanner(s);
char d = <statement>
```

QUESTION 12

What is output by the code segment to the right?

A. 10 4
B. 10 5
C. 11 3
D. 11 4
E. 9 6

```
String t = "";
String s = "state of texas";
do{
 int x = s.length()/2;
 t+=s.substring(x,x+1);
 s=s.substring(0,x)+s.substring(x+1);
}while(t.length()<10);
out.println(t.length()
      + " "+s.length());
```

QUESTION 13

What is output by the code segment to the right?

A. 4 B. 10 C. 20 D. 10true E. 20false

```
int x=2,y=5,z=7;
x+=x*y&z;
out.println(x);
```

QUESTION 14

What is output by the code segment to the right?

A. truetruetruetrue
B. truefalsetruefalse
C. truetruefalsefalse
D. truefalsefalsefalse
E. falsefalsefalsefalse

```
Integer x = 100;
Integer y = 100;
Integer z = new Integer(100);
out.print(x==y);
out.print(x==z);
x+=30;
y+=30;
out.print(x==y);
out.println(x==z);
```

QUESTION 15

What is output by the code segment to the right?

A. `[2, 5]` B. `[3, 5, 7]`
C. `[5, 7, 2]` D. `[3, 5, 7, 2]`
E. There is no output due to an error.

```
int [] list={9,3,5,7,2,4,1,6};
ArrayList<Integer> aList = new
ArrayList<Integer>();
for(int x:list)   aList.add(x);
out.println(aList.subList(2,5));
```

QUESTION 16

Which of the choices below shows the correct order of values to fill the blanks in the code to the right so that the output value is true?

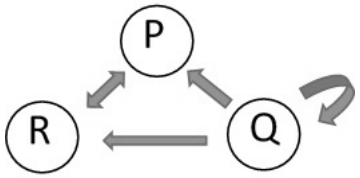
A. `1 3 6` B. `1 6 3` C. `3 1 6`
D. `3 6 1` E. `6 1 3`

```
String s = "animaniacs";
String t = "amaniacal";
out.println(s.regionMatches
              (___,t,___,___));
```

QUESTION 17

For the graph shown to the right, how many zeroes are in the adjacency matrix?

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



QUESTION 18

What is output by the code to the right?

A. `240` B. `255` C. `00000000`
D. `11110000` E. `11111111`

```
int x = 15, y = 4;
String s;
S = Integer.toBinaryString(x<<y);
out.println(s);
```

QUESTION 19

How many ordered triples make this boolean expression false?

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

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

QUESTION 20

Which choice below represents the ordered pair of values that would correctly fill in the blanks in the code to the right to generate a random integer in the range of 27 to 56, inclusive?

A. `27 29` B. `27 56` C. `29 28`
D. `29 30` E. `30 27`

```
Random r = new Random();
out.println(r.nextInt(___)+___);
```

QUESTION 21

Which of the following choices is NOT true about the Java PriorityQueue class?

A. Based on a heap data structure
B. Allows insertion of null elements
C. Head of the queue is the least object
D. Dynamic, able to grow and shrink as needed
E. Uses natural order of objects based on Comparable interface

QUESTION 22

Which of the following choices will correctly instantiate a new static two-dimensional array of doubles as shown below?

```
1.2 3.4
5.6 7.8 9.0
8.7 6.5
4.3
```

A. `double [] dubs = {1.2 3.4 5.6 7.8 9.0 8.7 6.5 4.3};`
B. `double [] dubs = {1.2, 3.4, 5.6, 7.8, 9.0, 8.7, 6.5, 4.3};`
C. `double [][] dubs = {1.2, 3.4}, {5.6, 7.8, 9.0}, {8.7, 6.5}, {4.3};`
D. `double [][] dubs = {{1.2, 3.4}, {5.6, 7.8, 9.0}, {8.7, 6.5}, {4.3}};`
E. All will work properly

QUESTION 23

Which of these descriptions regarding the least restrictive running times of TreeMap and HashMap methods is NOT correct?

A. HashMap get - constant time
B. HashMap put - constant time
C. TreeMap get - log(n) time
D. TreeMap put - log(n) time
E. All are correct

QUESTION 24

Which of the expressions below is NOT equivalent to the expression shown on the right?

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

A. $\overline{(\bar{A} * B)} + \overline{\overline{(A + \bar{B})}}$

B. $(\bar{A} + \bar{B}) + (A + \bar{B})$

C. $A + \bar{B}$

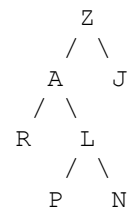
D. $\overline{(\bar{A} * B)} + (A + \bar{B})$

E. $\overline{(\bar{A} + \bar{B})} + (A + \bar{B})$

QUESTION 25

Each of four choices below matches one of the following traversals of the tree shown to the right: inorder, preorder, postorder, reverse order. Which choice does not match any of these traversals?

A. JZNLPAR B. ZAJRLPN C. ZARLPNJ
D. RAPLNZJ E. RPNLAJZ



QUESTION 26

What is output by the code segment to the right?

A. 8.1341 B. 8.135
C. 43.1 D. 53.1341
E. 91.1

```
int x = 5;
char a = '1';
String s = "34";
double d = 3.1;
out.println(x+d+s+a);
```

QUESTION 27

Which choice below correctly replaces <blank> in the code to the right?

A. String B. int C. char
D. double E. void

```
static <blank> mystR15(int x,int y){
  return Math.sqrt(x)+Math.cbrt(y);
}
//client code
out.println(mystR15(121,343));
```

QUESTION 28

Assuming the <blank> in the method to the right has been replaced correctly, what is output by the client code to the right?

A. 18 B. 18.0 C. 117
D. 117.0 E. There is no output due to an error.

QUESTION 29

Question omitted

QUESTION 30

Regarding the code to the right, which statement below best describes the implementation of class Two?

A. Class Two must implement the `alpha()` method.
B. Class Two must override the `beta()` method.
C. Class Two must implement `alpha()` and override `beta()`
D. Class Two is not required to do anything special.
E. This class structure is invalid due to an error in the code.

QUESTION 31

Assuming all is well with this class structure with any possible error having been corrected, and the alpha method having been defined to return the value 9, what is the output for the client code to the right?

A. `1.09B`
B. `1.0966`
C. `10.0B`
D. `76.0`
E. There is no output due to an error.

QUESTION 32

Which choice below is NOT be a valid set of statements for the implementation of Class Two in the code to the right?

A.
```
int alpha(){return 9;}
```

B.
```
String beta(){return "B";}
```

C.
```
int alpha(){return 9;}
String beta(){return "B";}
```

D.
```
int alpha(){return 10;}
String beta(){return "C";}
```

E. More than one of these is invalid.

```
abstract class One{
 double d = 1.0;
 abstract int alpha();
 String beta(){return "B";}
}
class Two extends One{
//    <implementation>
}
//client code
Two fer = new Two();
out.println(fer.d+fer.alpha()
           +fer.beta());
```

QUESTION 33

Which of the following choices best describes the quick sort algorithm?

A. Relies on a partition operation in which a pivot value is selected, and all elements smaller than the pivot are moved before it and all greater elements are moved after it.

B. Continuously divides a list into halves until each half is only one element, and then builds the halves back in sorted order until the entire list is sorted.

C. Works by taking elements from the list one by one and placing them in their correct position into a new sorted list.

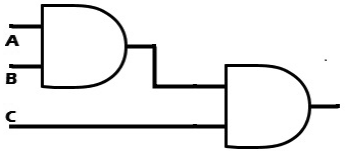
D. Starts at the beginning of the array and finds the best value for each position, swaps it with the value in the current position, and repeats these steps for the remainder of the list.

E. Starts at the beginning of the data set and compares the first two elements, and if the first is greater than the second, it swaps them. It continues doing this for each pair of adjacent elements to the end of the data set. It then starts again with the first two elements, repeating until no swaps have occurred on the last pass.

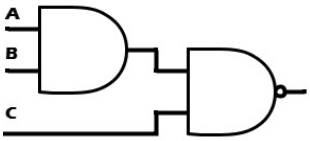
QUESTION 34

Which diagram below represents the logical statement on the right? $A + B + C$

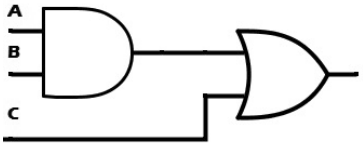
A.



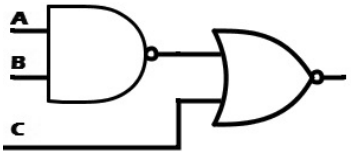
B.



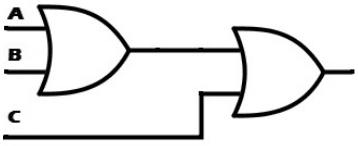
C.



D.



E.



QUESTION 35

Which of the following choices represents the decimal equivalent of the two's complement binary value 10011101?

A. -99 B. -101 C. -103 D. -105 E. -107

QUESTION 36

What is output by the code segment shown?

A. 4 B. 5 C. 6
D. 7 E. 8

```
LinkedList<Integer> list = new
LinkedList<Integer>();
list.add(4);
list.add(5);
list.addFirst(6);
list.addLast(7);
list.add(8);
list.pop();
out.println(list.pop());
```

QUESTION 37

What is output by the code segment to the right?

A. 40 B. 41 C. 63
D. 64 E. There is no output due to an error.

```
String s = "";
for(int x=1;x<40;x*=2)
      for(int y=1;y<=x;y++)
            s+='*';
out.println(s.length());
```

QUESTION 38

What is output by the code to the right?

A. 1 2 3 4 5 7 7 9
B. 1 5 9 7 2 4 3 7
C. 2 3 4 7 7 9 1 5
D. 9 7 1 5 2 4 3 7
E. 9 2 4 5 7 3 1 7

```
int [] list = {9,7,5,2,4,3,1,7};
Arrays.sort(list,1,5);
for(int x:list)
   out.print(x+" ");
```

QUESTION 39

Free Response Question:
Convert the expression below into the equivalent postfix expression.

- + / S A M * ^ S U ^ N G

QUESTION 40

Free Response Question:

Find f(3,5) using the function definition shown below.

```
f(x,y) = x              when y = 1
       = x * f(x,y-1)   when y > 1

f(3,5) = __________
```

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 Region 2015

1)	C	11)	B	21)	B	31)	C
2)	A	12)	A	22)	D	32)	B
3)	D	13)	A	23)	E	33)	A
4)	D	14)	D	24)	B	34)	E
5)	B	15)	C	25)	B	35)	A
6)	B	16)	C	26)	A	36)	A
7)	A	17)	C	27)	D	37)	C
8)	E	18)	D	28)	B	38)	E
9)	B	19)	C	29)	omitted	39)	SA/M+SU^NG^*-
10)	D	20)	E	30)	A	40)	243

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.

This page intentionally left blank

Explanations:

1. 74_{16} + 1101010_2 = 116_{10} + 106_{10} = 222_{10} = 336_8 = DE_{16} = 11011110_2
2. `21/3*5+8 = 7*5+8 = 35+8 = 43`
3. The correct labels for each part are: `System`-class, `out`-object, `println`-method, `"Hello"`-parameter
4. Each "s" is replaced by "th", not just the first one. The capital "S" is not considered.
5. In this expression, the second part of the expression where q is the only variable is the only part that really matters since q is true and using it with OR makes the entire expression true.
6. The log base 10 of 1000 is 3.0 (log10 returns a double) since 10 to the power of 3 is 1000.
7. The integer value of the character `'d'` is 100 (`'a'` is 97), therefore 100 - 33.3 is 66.7.
8. The sequence of values for a is: 0, 1(0+1), 4(1-2+5), 9(4+5), 16(9-1+8), 32(16+16), 33(32+1), 36(33-2+5), 41(36+5), 80(41-1+40), 160(80+80)
9. The sequence of values for a is: 100, 66, 44, 29, 19, 12, 8. Decimal division is indeed occurring, but the autocast feature of the /= operator converts the result back to an integer.
10. Here are the contents after the first reassignment statement - 5 3 1 3 2 - and here are the final contents - 5 2 1 3 2. The 4 is no longer in the list.
11. Only statements I and III WILL work with this code...nextChar is not a valid Java input statement.
12. Here is the sequence of the contents of each string throughout the loop.

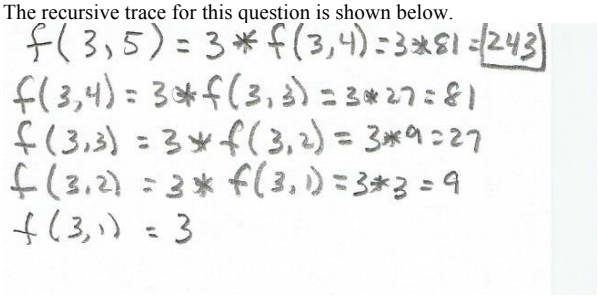
```
"" "state of texas"
"f" "state o texas"
"fo" "state  texas"
"fo " "state texas"
"fo  " "statetexas"
"fo  t" "stateexas"
"fo  te" "statexas"
"fo  tee" "statxas"
"fo  teet" "staxas"
"fo  teetx" "staas"
"fo  teetxa" "stas"
```

13. This expression involves operators from lines 3, 8, and 14 of the Java Operator Precedence chart. The 2*5 occurs first, followed by 10&7 (bitwise 1010_2 AND 111_2 ==> 0010_2), producing 2_{10}, which is then added to x resulting in the value 4 being output.
14. This problem involves the concept of common memory for Integer objects, in which any values between -127 and 128 directly assigned to Integer objects are stored in common memory, but any values constructed with the new operator are in separate memory. At first, x and y are indeed equal in the fact that they are referencing the same common memory value 100, but x and z are not since z was constructed with the new operator. Once both values are increased to 130, common memory is no longer involved, and both objects are in different memory locations.
15. The sublist method returns a view of the portion of the list between the specified fromIndex, inclusive, and toIndex, exclusive, working in a similar way to the String substring method.
16. The three parameters represent the start of the current String object (3 - "m"), the start of the parameter String (1 - "m"), and the length of the region (6) to be considered in both strings.
17. Below is the adjacency matrix for this graph, which contains 5 ones and 4 zeroes. The 1 in the top row at (P,R) indicates there is a direct one-hop path from vertex P to R, and likewise for the other 1s. A zero indicates there is no direct path.

```
  PQR
 P001
 Q111
 R100
```

18. 15 (00001111) left shift 4 becomes 240(11110000).
19. The four ordered triples that make this expression false are: 000, 001, 101, 111.
20. The formula for the first number is: hi-lo+1 (56-27+1), and the second number (offset) is always the low value(27), which means in this case there are 30 values in the range, starting with 27.
21. Since null elements are not comparable (have no natural ordering process), they cannot be inserted into a heap data structure which requires ordering to function.
22. Two layers of brackets {} and commas are required to establish the 2D array structure in a static assignment statement, the outside layer to indicate the rows, the inside layer the columns.
23. See the JDK Docs for more complete explanations for TreeMap and HashMap
24. Each choice, except for B, is a step in the simplification process. Choice A is the first step after breaking the top overbar using DeMorgan's law, followed by choice D as the double negative over the second parentheses disappears. Choice E follows as the overbar over the first term is broken, followed by $(A + \overline{B}) + (A + \overline{B})$ when the double negative over the first A goes away, and finally choice C where the two like terms merge together into one term.
25. Choice B is not one of the four traversals listed, but is the order in which the elements would be stored in a 1D array representation of this binary tree.
26. The evaluation sequence is as follows: 5 + 3.1 = 8.1, 8.1+"34" = "8.134", and finally "8.134" + '1' = "8.1341".
27. Since the two Math methods sqrt and cbrt return double values, the value produced by the return expression for this method is a double value, therefore the return data type should be **`double`**.
28. The square root of 121 (11.0) plus the cube root of 343 (7.0) equals 18.0.
29. A TreeSet uses the **compareTo** method to determine element duplication as well as order since storage in natural ascending order is a feature of this collection.
30. When a class extends an abstract class, any abstract method in the abstract class must be implemented. Any other method in the abstract class MAY be overridden, but that is not required.
31. The evaluation of the expression first mathematically adds 1.0 and 9 with a result of 10.0, which is then concatenated to "B" to create the String value "10.0B".

32. In the implementation of Class Two, since the alpha method is abstract in Class One, it must be defined in Class Two. Choice B only overrides beta(), which is optional. Choices A, C, and D are all valid.
33. B describes the merge sort, C the insertion sort, D the selection sort, and E the bubble sort.
34. Choice A is A $A * B * C$, choice B is $\overline{(A * B) * C}$, choice C is $A * B + C$, and choice D is $\overline{\overline{A + B} + C}$
35. 10011101 converts back to 01100011, which is the value 99, hence the original bit string is equivalent to -99.
36. Here are the contents of this list after each command in the code: [4], [4, 5], [6, 4, 5], [6, 4, 5, 7], [6, 4, 5, 7, 8], [4, 5, 7, 8], [5, 7, 8], with the 4 being popped at the end.
37. The outside x loop values are 1, 2, 4, 8, 16, and 32. The inside loop goes from 1 to each of those values, adding a star to the string each time. The length of the string is simply the sum of all of those termination values, which is 63.
38. The values from positions 1 to 4 (7,5,2,4) in the list are sorted, with the others remaning in their original places.
39. The equivalent infix expression for this prefix expression is
S / A + M - S ^ U * N ^ G, which then converts to the equivalent postfix version:
S A / M + S U ^ N G ^ * -
40. The recursive trace for this question is shown below.



University Interscholastic League
Computer Science Competition

Number 154 (State - 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 choice below represents the value of X in the equation shown?

$$10000_2 * X_{16} - 16_8 = 226_{10}$$

A. 15 B. 0.0242 C. F D. 0.024 E. 14

QUESTION 2

What is the result of the expression shown?

A. `20` B. `22` C. `26` D. `38` E. `92`

```
out.println(6 + 4 * 10 - 8);
```

QUESTION 3

How many line feeds are created by the output statement below?

```
System.out.printf("Hello\nall\\nyou%nbrilliant\npeople!");
```

A. 1 B. 2 C. 3 D. 4 E. 5

QUESTION 4

What is output by the code segment below?

```
String s = "moonlight";
String t = "sunlight";
int b = s.length();
int a = t.length();
out.println(s.substring(0,a).compareTo(t.substring(b/2)));
```

A. `1` B. `-1` C. `4` D. `-4` E. 0

QUESTION 5

What is output by the code segment to the right?

A. `false` B. `true`

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

QUESTION 6

Which of the choices below fills the blank so that output of this code segment is `true`?

A. 2 B. `2*Math.sqrt(2)`
C. `2*Math.cbrt(2)` D. `4`
E. 8

```
double b = ___;
double c = 4;
out.println(Math.abs(Math.sin
    (Math.PI/6)-b/c)<0.00001);
```

QUESTION 7

What is output by the code to the right?

A. `6` B. `7` C. `54` D. `55`
E. There is no output due to an error.

```
long j = 2;
char a = '9';
out.println(a+=~j);
```

QUESTION 8

Which of the colors in the code segment to the right will be output the fewest number of times?

A. red B. green C. blue D. black

E. tie between two colors

```
for(int x=-5;x<11;x++){
if(x>0){
  if(x<9)  out.println("red");
  else     out.println("green");
}
else
  if(x<0)
  out.println("blue");
if(x==0)   out.println("black");
}
```

QUESTION 9

Which of the choices below is NOT an output value of the code segment to the right?

A. 13 B. 40 C. 121 D. 363

E. There is no output due to an error

```
for(int x=1;x<400;x*=3,x++)
  out.print(x);
```

QUESTION 10

What is the final value output by the code segment to the right?

A. 0.0 B. 1.0 C. 1.5
D. 2.0 E. 2.5

```
double [] dip = new double[5];
for(int x=1;x<5;x++)
  dip[x]=x/2;
for(double d:dip)
  out.print(d+" ");
```

QUESTION 11

What is output by the code segment to the right?

A. 168 B. 1 6 8 C. 123468910
D. 1 2 3 4 6 8 9 10
E. There is no output due to an error.

```
String s="1 2 3 4 5 6 7 8 9 10";
Scanner f = new Scanner(s);
f.useDelimiter("[57]");
while(f.hasNext()){
 Scanner g = new
    Scanner(f.next());
 out.print(g.next());
}
```

QUESTION 12

What is output by the code segment to the right?

A. ABCDE B. abcde C. EDCBA D. edcba
E. There is no output due to an error.

```
String s = "";
for(int x=65;x<70;x++)
      s=(char)x+s;
out.println(s);
```

QUESTION 13

What is output by the code segment to the right?

A. 1 B. 7 C. 20 D. 25
E. There is no output due to an error.

```
int x=2,y=5,z=7;
out.println(x|y==z?y<<x:z>>x);
```

QUESTION 14

What is the final output value in the code segment to the right?

A. 127 B. 128 C. -127 D. -128

E. There is no final output value due to an infinite loop

```
byte x = 0;
while(x>=0)
      out.println(++x);
```

QUESTION 15

What is output by the code segment to the right?

A. `true[9, 7, 4, 6]`
B. `false[9, 7, 4, 6]`
C. `true[]`
D. `false[]`
E. There is no output due to an error.

```
int [] list1={9,5,7,4,6};
int [] list2={5,2,1};
ArrayList<Integer> List1 = new
ArrayList<Integer>();
for(int x:list1) List1.add(x);
ArrayList<Integer> List2 = new
ArrayList<Integer>();
for(int x:list2) List2.add(x);
out.print(List1.removeAll(List2));
out.println(List1);
```

QUESTION 16

The method to the right calculates interest compounded annually. What is output by the client code shown?

A. 1000.0 B. 1100.0 C. 1210.0
D. 1331.0 E. 1464.1

```
static double mystS15(int x,int y,
int z){
      double w = x;
      for(int a=1;a<=z;a++)
            w+=w*y/100;
      return w;
}
//client code
out.println(mystS15(1000,10,2));
```

QUESTION 17

Which of the loop structures below correctly replaces the bolded portion of the method to the right in order to compound monthly with the same effective annual interest rate?

A.
```
for(int a=1;a<=z*12;a++)
w+=w*y/100/12;
```
B.
```
for(int a=1;a<=z;a++)
w+=w*y/100/12;
```
C.
```
for(int a=1;a<=z*12;a++)
w+=w*y/100;
```
D.
```
for(int a=1;a<=z/12;a++)
w+=w*y/100/12;
```
E.
```
for(int a=1;a<=z*12;a++)
w+=w*y/100*12;
```

QUESTION 18

How many multiples of 3 are output by the code segment to the right?

A. 6
B. 7
C. 8
D. 9
E. There is no output due to an error.

```
int [][] list = new int[4][0];
for(int r=0;r<list.length;r++){
   list[r] = new int[(r+1)*2];
   for(int c=0;c<list[r].length;c++)
      list[r][c]=(r+1)*(c+1);
}
for(int [] nums:list)
   for(int d:nums)
      out.print(d+" ");
```

QUESTION 19

Which of the ordered triples listed below makes the Boolean expression on the right true?

A. 000
B. 001
C. 010
D. 100
E. 110

$$\overline{\left(\bar{A} \oplus \bar{B}\right)} * C$$

QUESTION 20

See Reference Page #1 for Questions 20-23 at the end of this test.

Using the code on Reference Page #1, which choice below, if necessary, is a correct replacement for <code> in line 1?

A. No code is necessary here B. `extends Comparable` C. `extends Comparator`
D. `implements Comparable` E. `implements Comparator`

QUESTION 21

Assuming the issue in Question 20 is resolved correctly according to the given choices, what is the correct replacement for <className> in line 11 of the code?

A. `Object` B. `Team` C. `ArrayList` D. `ClassTeam`

QUESTION 22

In lines 1-10 of the code, which line, if any contains an issue contrary to the specification of this process as indicated in the problem description?

A. 7 B. 8 C. 9 D. 10 E. There is no issue in this section of code

QUESTION 23

Assuming any issues in the code have been resolved so that the code behaves as described, what is the output of the client code shown?

A.
```
three 9:0 2 3 4
four 9:0 3 3 3
one 9:1 2 3 4
two 9:1 2 3 4
seven 9:2 2 3 4
six 10:0 2 3 5
five 10:1 2 3 5
```

B.
```
five 10:1 2 3 5
six 10:0 2 3 5
seven 9:2 2 3 4
one 9:1 2 3 4
two 9:1 2 3 4
three 9:0 2 3 4
four 9:0 3 3 3
```

C.
```
five 10:1 2 3 5
six 10:0 2 3 5
seven 9:2 2 3 4
two 9:1 2 3 4
one 9:1 2 3 4
four 9:0 3 3 3
three 9:0 2 3 4
```

D.
```
five 10:1 2 3 5
six 10:0 2 3 5
seven 9:2 2 3 4
two 9:1 2 3 4
one 9:1 2 3 4
three 9:0 2 3 4
four 9:0 3 3 3
```

QUESTION 24

An algorithm with a least restrictive running time of $O(N\log_2 N)$ processes 1024 data elements in 5 seconds. How many data elements can be processed by the same algorithm in 24 seconds?

A. 2048 B. 4096 C. 4915 D. 5120 E. 49152

QUESTION 25

What is the output of the code to the right?

A. 1 B. 25 C. 57 D. 58

E. There is no output due to an error

```
int x = 33;
int y = 25;
int z = x&y;
out.println(z);
```

QUESTION 26

What is output by the code segment to the right?

A. 15 B. 19 C. 20

D. 21 E. 25

```
String s="";
for(int x=-2;x<=2;x++){
      int y=x;
      while(y<7){
            s+="*";
            y+=2;
}}
out.println(s.length());
```

QUESTION 27

See Reference Page #2 at the end of this test packet for the code that relates to Questions 27 and 28.

Below are the first few lines of output by the client code on Reference Page #2. What will be the next line of output?

```
MergeSortHelper 1
MergeSortHelper 2
MergeSortHelper 3
MergeSortHelper 4
MergeSortHelper 5
merge:0 1
MergeSortHelper 6
```

A. `MergeSortHelper 7` B. `merge:0 2` C. `merge:1 2` D. `merge:0 3`

E. Not possible to determine

QUESTION 28

What value follows the last output of this code that begins with "**`MergeSortHelper`** "?

A. 9 B. 10 C. 11 D. 12 E. Not possible to determine

QUESTION 29

Given a dictionary with 2000 pages containing words and definitions, using the binary search process, what is the most number of search steps it would take to **find the page** on which any given word and its definition is located, or that it is not found in the dictionary at all?

For example, the first search step would be to look on page 1000, then proceed with the standard binary search process from there.

A. 10 B. 11 C. 12 D. 13 E. Cannot be determined

QUESTION 30

Using standard math operator order (with **^** being the exponent operator), which of the expressions below are equivalent to this expression?

```
A D V ^ - A N ^ C * E D ^ / +

I.   A - D ^ V + A ^ N * C / E ^ D
II.  - + A ^ D V / * ^ A N C ^ E D
III. A + D ^ V - A ^ N * C / E ^ D
IV.  + - A ^ D V / * ^ A N C ^ E D
```

A. I and II B. III and IV C. II and III D. I and IV

QUESTION 31

What is output by the code to the right?

A. -5

B. -20

C. 180

D. 200

E. 220

```
public static int f(int x){
  if(x>50)return f(x/2)+2*x;
  if(x>0)return f(x-10)-3;
  return -5;
}
//client code
out.print(f(100));
```

QUESTION 32

What is output by the code to the right?

A. 1911 B. 19.011 C. 910.11

D. 91011 E. 20.0

```
int m = 9;
double n = 10;
String p = "11";
out.println(m+n+p);
```

QUESTION 33

What is the 5th value output by the code to the right?

A. 1 B. 2 C. 3

D. 5 E. 7

```
int [] list = {2,7,5,1,3,7,0};
PriorityQueue<Integer> p = new
PriorityQueue<Integer>();
for(int x:list)
      p.add(x);
while(!p.isEmpty())
      out.println(p.remove());
```

QUESTION 34

Which expression below represents the diagram on the right?

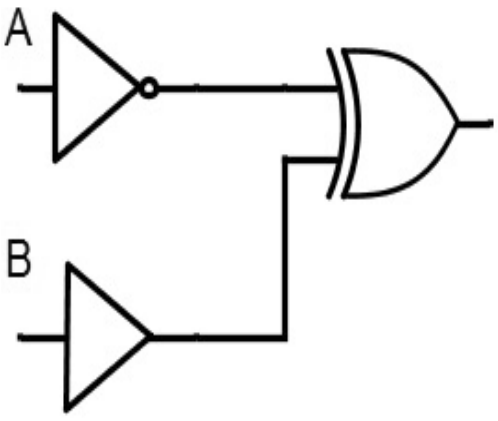
A. $\bar{A} \oplus B$

B. $A \oplus B$

C. $\bar{A} + B$

D. $A + B$

E. $\bar{A} * B$



QUESTION 35

Using the code to the right, if the following pairs of values were each assigned to x and y, respectively, how many would result in the output "CS"?

`3 2  1 7  5 3  4 -3  4 5  -6 5`

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

```
int x=<value>,y=<value>;
String s =
y%x==1?"UIL":x%y==1?"CS":"STATE";
out.println(s);
```

QUESTION 36

What is output by the code to the right?

A. xtr B. rd C. e D. ao

E. A blank is output due to an empty string in that position.

```
String p = "[aeiou]+";
String s = "extraordinary";
out.println(s.split(p)[1]);
```

QUESTION 37

Which choice below represents the simplest form of the expression shown to the right?

A. $\bar{A}$ B. B

C. $\bar{B}$ D. $A * \bar{B}$

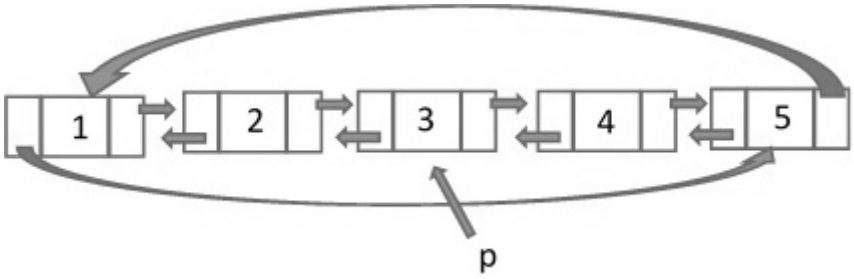
E. $A * \bar{B} + \bar{B} + \bar{A} * \bar{B}$

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

QUESTION 38

Consider the diagram of the doubly linked list shown below, each node with one integer data field and two pointers, **next** pointing right and **back** pointing left, as in the class definition shown, and assuming that sufficient code has produced the list in the diagram with DLLNode object p referencing the list as indicated. What is the output of the client code shown?

```
class DLLNode{
      public int data;
      public DLLNode next, back;
}
//client code
out.println(p.next.next.data + p.back.back.back.data);
```



A. 2 B. 4 C. 5 D. 7 E. 10

QUESTION 39

Free Response Question:

What is the decimal equivalent for the 8-bit binary value 10101001?

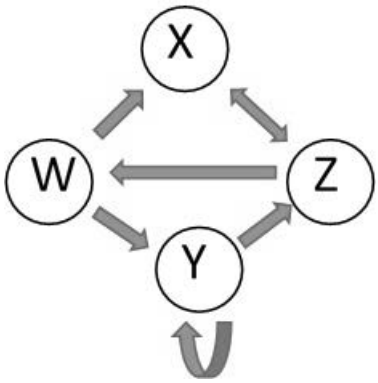
QUESTION 40

Free Response Question:

In the graph shown below, there are seven paths of length 1: WX, WY, XZ, ZX, YY, YZ, and ZW.

How many different paths of length <u>THREE</u> are there?

For example, WXZW, XZWX and WYYY are all different paths of length 3.



NO TEST MATERIAL ON THIS PAGE

Reference Page #1 for Questions 20-23.
In the rules of a certain contest similar to UIL, a team can have three or four members, but only the top three scores are considered for the team score. In case there is a team score tie, the following procedure is used to break that tie between two teams. The higher score of the fourth place member of the team will be used to break the tie. Should two or more contestants who are the fourth place members of the team have the same score, then an unbreakable tie will be declared. If one of the teams does not have a fourth person, it loses the tie breaker. If neither team has a fourth member and the team score is tied, it is an unbreakable tie. For the purpose of this scenario, a zero score means no member participated.

The code below defines the Team class, and the **compareTo** method attempts to implement this complex tie break procedure.

```
1 class Team <code>{
2      String team;
3      int [] scores = new int [4];
4      int teamScore;
5      public Team(String t,int a, int b, int c, int d){
6             team = t;
7             scores[0]=a;scores[1]=b;scores[2]=c;scores[3]=d;
8             teamScore=a+b+c+d;
9             Arrays.sort(scores);
10            teamScore-=scores[3];
       }
/*
precondition: parameter o references a Team object
postcondition: method returns 1 for a team that has a higher team score, or that wins the
tiebreaker, returns -1 for a team that has a lower team score, or that loses the tiebreaker, or
returns 0 for an unbreakable tie
*/
11     public int compareTo(<className> o){
12            Team u = (Team)o;
13            if(this.teamScore<u.teamScore)return -1;
14            else if(this.teamScore>u.teamScore)return 1;
15            else if(this.scores[0]>0&&u.scores[0]>0)
16                     if(this.scores[0]<u.scores[0])
17                           return -1;
18                    else if(this.scores[0]>u.scores[0])
19                           return 1;
20                    else return 0;
21            else
22            if(this.scores[0]>0&&u.scores[0]==0||
23               this.scores[0]==0&&u.scores[0]>0)
24                   if(this.scores[0]>0)
25                          return 1;
26                   else   return -1;
27            else return 0;
28     }
29     public String toString(){
30            String t = team+" "+teamScore+":";
31            for(int x:scores)
32                   t+=x+" ";
33            return t;
       }
}
//client code
String s;
s = "one 1 2 3 4 two 1 2 3 4 three 0 2 3 4 four 3 0 3 3 five 1 2 3 5 six 0 2 3 5 seven 2 2 3 4";
ArrayList<Team> list = new ArrayList<Team>();
Scanner f = new Scanner(s);
while(f.hasNext())
       list.add(new Team(f.next(),f.nextInt(),f.nextInt(),f.nextInt(),f.nextInt()));
Collections.sort(list);
Collections.reverse(list);
for(Team l:list)
       out.println(l);
```

Reference Page #2 for Questions 27-28.

```
public static void mergeSort(int[] list){
 int n = list.length;
 int[] temp = new int[n];
 mergeSortHelper(list, 0, n-1, temp);
}
private static void mergeSortHelper(int[] list,int front,int back,int[] temp){
 out.println("MergeSortHelper "+a++);
 if (front < back){
 int mid = (front + back) / 2;
 mergeSortHelper(list, front, mid, temp);
 mergeSortHelper(list, mid + 1, back, temp);
 merge(list, front, mid, back, temp);
 }
}
private static void merge(int[] list,int front,int mid,int back,int[] temp){
      int i = front, j = mid + 1, k = front;
      out.println("merge:"+front+" "+back);
      while (i <= mid && j <= back){
            if(list[i] < list[j])
                  temp[k++] = list[i++];
            else
                  temp[k++] = list[j++];
      }
      while(i <= mid)
             temp[k++] = list[i++];
      while(j <= back)
             temp[k++] = list[j++];
      for(int x=front;x<=back;x++)
            list[x]=temp[x];
}
//client code
int [] list = {8, 2, 4, 7, 5, 1};
mergeSort(list);
```

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 State 2015

1)	C	11)	A	21)	A	31)	C
2)	D	12)	C	22)	D	32)	B
3)	C	13)	E	23)	C	33)	D
4)	C	14)	D	24)	B	34)	A
5)	B	15)	A	25)	A	35)	B
6)	A	16)	C	26)	B	36)	A
7)	A	17)	A	27)	B	37)	C
8)	D	18)	D	28)	C	38)	E
9)	D	19)	B	29)	B	39)	-87
10)	D	20)	D	30)	D	40)	21

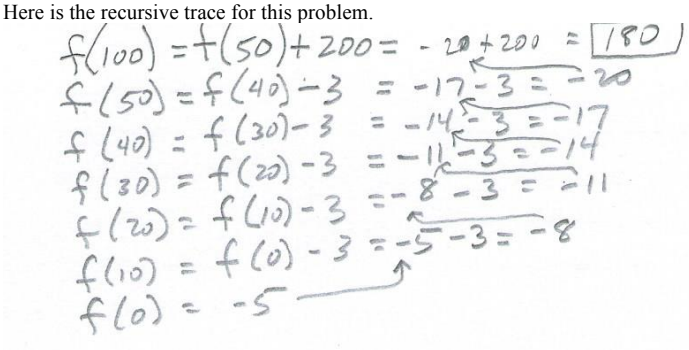
Note to Graders:

- 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.

This page intentionally left blank

Explanations:

1. The equation $\mathbf{10000_2X_{16} - 16_8 = 226_{10}}$ translates into the base 10 equation **16X - 14 = 226**, which transforms to **16X = 240**, and finally **X=15**, which is the value **F** in base 16.
2. `6 + 4 * 10 - 8` ➔ `6 + 40 - 8` ➔ `46 - 8` ➔ `38`
3. The three line feeds are created by the `\n`, `%n`, and `\n`. The `\\n` does not create a line feed, but instead outputs "`\n`". Also, unlike the println statement, there is no automatic line feed created by the printf method.
4. The string "moonligh" is 4 characters ***greater*** than the string "ight", since the letter 'm' is four places beyond the letter 'i' in the alphabet.
5. Since P XOR Q is false when both values of P and Q are true, the NOT value is true.
6. The sin ratio of a 30 degree angle (PI/6) is 0.5 (opposite over hypotenuse), which requires the value of b to be 2 since b is the shortest side of 30-60-90 triangle whose hypotenuse is 4. The actual output of Math.sin(Math.PI/6) is 0.49999999999999994, which is virtually 0.5, but using == would output false, thus requiring the subtraction expression as shown to test for virtual equality.
7. '9' += ~2 ➔ '9' += -3 ➔ '6'. The += preserves the char because of autocasting. **Note:** The ~ (complement) operator gives the opposite of a value minus 1, therefore the opposite of 2 is -2, minus 1 is negative 3. Conversely, ~-3 produces 2 (opposite of -3 is 3, minus 1 is 2).
8. The color blue is output 5 times (-5 to -1), black once(0), red 8 times(1 through 8), and green twice(9 and 10).
9. The output sequence is: 1 4 13 40 121 364.
10. The output values are: 0.0 0.0 1.0 1.0 2.0
11. The code loops three times using the strings "`1 2 3 4 `", "` 6 `", and "` 8 9 10`", taking the first non-space element of each string, resulting in an output of "168"
12. Since the ASCII values of "ABCDE" are 65-69, these characters are accumulated in the string, but in reverse order.
13. In order for this code to work properly, a parentheses must be placed into the expression `out.println(`**`(x|y)`**`==z?y<<x:z>>x);` since **==** has a higher order of precedence than **|**, and bitwise OR cannot take a boolean value as an operand along with an integer.
14. Since the increment happens before the output and after the check, the last few outputs of this code are: 126, 127, -128, with the byte value wrapping around to the minimum value to end the loop.
15. Although the removeAll method only removes one item, the return value is still true since something was removed, and the contents of List1 is the same, less the 5.
16. The initial principal amount is 1000.0, compounded annually at 10% for 2 years, which after the first year is 1100.0 (1000+100), and after two years 1210.0 (1100+110).
17. The correct adjustment increases the number of loop iterations by 12, and divides by 12 the interest compounded.
18. The complete output of the code is: 1 2 2 4 **6** 8 **3 6 9 12 15 18** 4 8 **12** 16 20 **24** 28 32, showing 9 multiples of 3.
19. If A and B are both false, their NOT values are both true, making the XOR value false, and the NOT XOR value true, which when ANDed with a true value of C causes the entire expression to be true. The triple 111 will also make this expression true.
20. Since this class uses the compareTo method, an undefined method of the Comparable interface, "implements Comparable" is required in the class header.
21. In this case, the required parameter type for the compareTo method is Object. Once inside the method, the object reference can be changed to something more specific.
22. Line 10 contains a logic error, or bug. The code should be **`teamScore-=scores[0];`** since the lowest team member's score is in position zero after the Arrays.sort statement, which arranges the scores in ascending order.
23. Team five wins over team six because it has a fourth member. Team seven wins the 5-way tie by virtue of the greater 4th score. Teams one and two tie, but have a fourth member, thus ranking higher than teams three and four, who also tied. However, since the Collections.sort did not change the relative positions of the tied teams, when the list is reversed, team two is listed before one, and four before three.
24. The equation setup is: $1024*10/5 = N\log_2 N/24$, which resolves to 4096*12/24, with an N value of 4096.
25. `33&25 ==> 100001&11001 ==> 000001 ==> 1`
26. The 19 values of y in the loop process are: `-2 0 2 4 6 -1 1 3 5 0 2 4 6 1 3 5 2 4 6`
27. The "front, back" sequence for the first five MergeSortHelper calls is: 0 5, 0 2, 0 1, 0 0, 1 1, at which time the first merge occurs for the first two elements in the list. MergeSortHelper 6 uses the front/back values 2 2, at which time the next merge occurs between the first list that was merged (containing elements in positions 0 and 1), and the single element at position 2, creating the output "merge: 0 2".
28. There are eleven different front/back splits in this process: 0 5, 0 2, 0 1, 0 0, 1 1, 2 2, 3 5, 3 4, 3 3, 4 4, and 5 5.
29. Since the binary search is a O(log N) process, the number of searches is the log base 2 exponent number that creates a value that equals or just exceeds 2000, which 11. 2^11 is 2048, therefore it will take at most eleven search steps to find the **page** on which the word exists, or that the word is not found.
30. The original expression is postfix form. Choice I is the infix equivalent, and choice IV is the prefix equivalent.
31. Here is the recursive trace for this problem.



32. Since the values 9 and 10.0 are added mathematically, that result is 19.0, which is then concatenated with "11", resulting in "19.011".
33. The complete output order is: 0 1 2 3 5 7 7.
34. $\bar{A} \oplus B$ is the correct expression, with the bar over the A indicating NOT, and the circle-plus operator for XOR.
35. The output values for each pair are: `3 2==> CS, 1 7==> CS, 5 3==> STATE, 4 -3==> CS, 4 5==> UIL, -6 5==> STATE.`
36. The split pattern "`[aeiou]+`" means split on any sequence of those letters, wherever they occur in the string, in whatever order. Since the first split occurs at the `'e'`, there is an empty string at the front, which means position 1 contains "xtr".

37. $\overline{\overline{A} * B} * (\overline{B} + \overline{A})$ simplifies to $(A + \overline{B}) * (\overline{B} + \overline{A})$ using DeMorgan's law and the Double Negative law, then to $A\overline{B} + \overline{B} + \overline{A}\overline{B}$ (choice E) using FOIL and the complement law for AND, and finally to $\overline{B}$ using the absorption law.
38. Since both pointer expressions are referencing the value 5 at the end of the list, the sum of 5 and 5 is 10. **Note**: the position of p does not change in either expression.
39. Using the two's complement conversion process, 10101001 converts back to 01010111, which is the value 87, hence the original bit string is -87.
40. There are 21 different paths of length 3: WXZW, WXZX, WYYY, WYZX, WYYZ, WYZW, XZXZ, XZWY, XZWX, YYYY, YYYZ, YYZX, YZXZ, YZWX, YZWY, YYZW, ZXZX, ZXZW, ZWYZ, ZWYY, and ZWXZ. These can be found using the "brute force" method by careful examination, or using the matrix multiplication process as shown below.

Using the matrix multiplication method produces the following matrices:
Adjacency matrix M1
(7 paths of length 1)

```
  WXYZ
W 0110
X 0001
Y 0011
Z 1100
```

M1*M1 = M2 (12 paths of length 2)

```
  WXYZ
W 0012
X 1100
Y 1111
Z 0111
```

M1*M2 = M3 (21 paths of length 3)

```
  WXYZ
W 2211
X 0111
Y 1222
Z 1112
```