University Interscholastic League

Computer Science Competition

Number 134 (District 2 - 2012)

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. **All provided code segments are intended to be syntactically correct, unless otherwise stated. Ignore any typographical errors and assume any undefined variables are defined as used.**

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.

10) Assume that any necessary import statements for standard Java packages and classes (e.g. `.util`, `ArrayList`, etc.) are included in any programs or code segments that refer to methods from these classes and packages.

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.

QUESTION 1

What is the sum of $D0A_{16}$ and 238_{16}?

A. $F42_{16}$ B. $AD2_{16}$ C. $E48_{16}$ D. $F48_{16}$ E. $E56_{16}$

QUESTION 2

What is output by the code to the right?

A. 4 B. 28 C. 56

D. 77 E. 84

```
int x = 4;
int y = x + 3 * (4 + x);
System.out.print(y);
```

QUESTION 3

What is output by the code to the right?

A. 0 B. 9 C. 10

D. 24 E. 27

```
int val = 0;
for(int i = 10; i > 1; i--)
  val += 3;
System.out.print(val);
```

QUESTION 4

What is output by the code to the right?

A. 13 B. 12 C. 7

D. 6 E. -1

```
String c1 = "AABABBAAABBBAAABB";
int res = c1.indexOf("AAA", 8);
System.out.print(res);
```

QUESTION 5

What is output by the code to the right?

A. 1 2 B. 3 2 C. 65 12

D. 2 1 E. 2 3

```
String[] st = {"AB", "A", "DAD", "12"};
System.out.print(st[3].length());
System.out.print(" " + st[1].length());
```

QUESTION 6

What is output by the code to the right?

A. 158 B. 96 C. 64

D. 32 E. 19

```
int x1 = 13;
int y1 = 19;
int z1 = 2 * x1 - y1 + 2 * y1 - x1;
System.out.print(z1);
```

QUESTION 7

How many combinations of values for the `boolean` variables `p`, `q`, and `r` will result in `s` being set to `true`?

A. 1 B. 2 C. 3

D. 4 E. 7

```
boolean p, q, r;
//code to initialize p, q, and r

boolean s = (p || q) && (!r && !q);
```

QUESTION 8

What is output by the code to the right?

A. `43.85` B. `13.85`

C. `47.7` D. `17.7`

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

```
double a3 = 7.7;
if(a3 / 2 > 3)
  System.out.print(1);
else
  System.out.print(4);
System.out.print(a3);
```

QUESTION 9

Given the `Student` and `Block` classes to the right, what is output by the following client code?

```
Student st1 = new Student(10);
st1.goodYear();
System.out.print(st1);
```

A. `credits: 6` B. `10`

C. `credits: 16` D. `6`

E. `credits: credits`

QUESTION 10

Given the `Student` and `Block` classes to the right, what is output by the following client code?

```
Student st2 = new Block(10, 4);
st2.goodYear();
System.out.print(st2);
```

A. `credits: 16` B. `credits: 14`

C. `credits: 10` D. `credits: 12`

E. There is no output due to a syntax error in the client code.

```
public class Student {
  private int credits;

  public Student(int c) {credits = c;}

  public void goodYear() {credits += 6;}

  public String toString() {
    return "credits: " + credits;
  }

  public void setCredits(int c) {
    credits = c;
  }
}

public class Block extends Student {
  private int ccs;

  public Block (int c, int ex) {
    super(c);
    ccs = ex;
  }

  public void goodYear() {
    setCredits(ccs + 8);
  }
}
```

QUESTION 11

What is output by the code to the right?

A. `189` B. `127` C. `62`

D. `31` E. `0`

```
int m = 127;
int n = 62;
System.out.print(m & n);
```

QUESTION 12

What is output by the code to the right?

A. `6.0` B. `5.0` C. `4.825`

D. `4.0` E. `3.15`

```
double m2 = 3.15;
double n2 = m2 / 2;
double o2 = Math.floor(n2) + Math.ceil(m2);
System.out.print(o2);
```

QUESTION 13

What is output by the code to the right?

A. `bigtallthin` B. `bigtall\nthin`

C. `big`
`tall\nthin` D. `big`
`tallnthin`

E. `big`
`tall`
`thin`

```
System.out.print("big");
System.out.println();
System.out.print("tall\nthin");
```

QUESTION 14

What is output by the code to the right?

A. `671.5` B. `+671.4` C. `671.4`

D. `(671.0)` E. `+671.5`

```
double t5 = 671.45;
System.out.printf("%+6.1f", t5);
```

QUESTION 15

What is returned by the method call `calc(-2, -1.5)`?

A. `-31.5` B. `-21.0` C. `-15.0`

D. `-13.5` E. `3.5`

```
public double calc(int x, double a) {
  x--;
  a -= 2 * x;
  return a * x;
}
```

QUESTION 16

What is output by the code to the right?

A. `16` B. `15` C. `11`

D. `10` E. `0`

```
String stars = "";
for(int i = 0; i < 5; i++)
    stars += "*";
stars += "*";
for(int i = 0; i < 10; i++)
    stars += "*";
System.out.print(stars.length());
```

QUESTION 17

What is output by the code to the right?

A. `2` B. `2.0` C. `4.0`

D. `4` E. `5.0`

```
double[] as = {0.5, -1.78, 2.21, 4.5000};
System.out.print( (int) as[3] );
```

QUESTION 18

What is output by the code to the right?

A. `0` B. `25` C. `25.0`

D. `32` E. `32.0`

```
int r = 0;
r += Math.pow(2, 5);
System.out.print(r);
```

QUESTION 19

Which of the following Java statements is equivalent to the formula to the right? `BMI`, `ma`, and `in` are variables of type `double`.

A. `BMI = 703 * ma / (in * in);`

B. `BMI = 703 * m * a / in / 2;`

C. `BMI = 703.0 * ma / (in ^ 2);`

D. `BMI = 703.0 * ma / in * 2;`

E. `BMI = 703 * m * a >> in >> in;`

$$BMI = \frac{703ma}{in^2}$$

QUESTION 20

What is the minimum value the code to the right will print out?

A. `0.0` B. `1.0` C. `2.0`

D. `3.0` E. None of A through D are correct.

```
double mys = 0.0;
double t = 0;
do {
  t = Math.random();
  mys++;
} while(t < 0.2);
System.out.print(mys);
```

QUESTION 21

What is output by the code to the right?

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

C. `false false`

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

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

```
Object obj1 = new ArrayList<String>();
Object obj2 = obj1.toString();
System.out.print(obj1 == obj2);
System.out.print(" " + obj1.equals(obj2));
```

QUESTION 22

Which of the following `boolean` expressions replaces **<*1>** in method `sort` so that the body of the `if` statement is executed if the `String` at index `m` is greater than the `String` at index `j`?

A. `w[m].compareTo(w[j]) > 0`

B. `w[m] > w[j]`

C. `w[m] < w[j]`

D. `w[m].compareTo(w[j]) < 0`

E. `Comparable.compareTo(w[m], w[j]) > 0`

Assume **<*1>** is filled in correctly.

QUESTION 23

What is output when the client code to the right is executed?

A. `[Z, a, MY, Z, b, p, BE]`

B. `[BE, MY, Z, Z, a, b, p]`

C. `[BE, MY, Z, Z, a, p, b]`

D. `[Z, Z, BE, MY, b, p, a]`

E. `[BE, MY, Z, Z, b, p, a]`

QUESTION 24

Which sorting algorithm does method `sort` implement?

A. insertion sort B. selection sort C. radix sort

D. quicksort E. heap sort

```
public void sort(String[] w) {
  for(int i = 0; i < w.length - 1; i++) {
    int m = i;
    for(int j = i + 1; j < w.length; j++)
      if( <*1> )
        m = j;
    if(m != i) {
      String t = w[i];
      w[i] = w[m];
      w[m] = t;
    }
    if(i == 4)
      System.out.print(Arrays.toString(w));
  }
}

// client code
String[] ds = {"Z", "a", "MY", "Z", "b",
                                    "p", "BE"};
sort(ds);
```

QUESTION 25

Which of the following is a valid Java identifier?

A. `+12` B. `5x` C. `(val)` D. `x_y12` E. More than one of A through D is correct.

QUESTION 26

What replaces **<*1>** in the method `search` to the right so that the output of the client code to the right is `[0, 2, 4, 8]`?

A. `m[i] = t` B. `m[i].equals(t)`

C. `m[i] == t` D. `m[i].compareTo(t)`

E. More than one of A through D is correct.

```
public ArrayList<Integer> search(
                    String[] m, String t) {
  ArrayList<Integer> r;
  r = new ArrayList<Integer>();
  for(int i = 0; i < m.length; i++)
    if(<*1>)
      r.add(i);
  return r;
}

// client code
String[] n = {"A", "N", "A", "J", "A",
                    "M", "AA", "AAA", "A"};
String sch = n[6].substring(1);
System.out.print(search(n, sch));
```

Assume **<*1>** is filled in correctly.

QUESTION 27

Which searching algorithm does method `search` implement?

A. binary B. heap

C. radix D. sequential

E. insertion

QUESTION 28

Which of the following can replace **<*1>** in the code to the right so that the code segment compiles without error?

I. `Collection<Integer>`
II. `List<Integer>`
III. `Object`

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

D. I and II only E. I, II, and III

```
<*1> list = new LinkedList<Integer>();
System.out.print(list.toString());
System.out.print(list.size());
```

QUESTION 29

What replaces **<*1>** in method mystery to the right to obtain the remainder of the integer division of `x` by `i`?

A. `rem` B. `~` C. `!`

D. `/` E. `%`

```
public int mystery(int x) {
  int num = 0;
  for(int i = 1; i <= x; i++)
    if( x <*1> i == 0 )
      num++;
  return num;
}

// client code
System.out.print(mystery(13));
System.out.print(" " + mystery(72));
```

Assume **<*1>** is filled in correctly.

QUESTION 30

What is output by the client code to the right?

A. `2 12` B. `13 72` C. `1 11`

D. `1 7` E. `0 10`

QUESTION 31

What is output by the code to the right?

A. `10001100` B. `01110011` C. `00000000`

D. `11110011` E. `11111111`

```
byte b1 = 12;
byte b2 = (byte) ~b1;
String bits = Integer.toBinaryString(b2);
bits = bits.substring(bits.length() - 8);
System.out.print(bits);
```

QUESTION 32

What is the average case order (Big O) of method `find` shown to the right, given the following kinds of `Maps`? `m.size()` = N and `can.length` = M. Pick the most restrictive correct set of answers.

	TreeMap	HashMap
A.	O(NM)	O(NM)
B.	O(NlogM)	O(NM)
C.	$O(MN^2)$	O(N)
D.	$O(M^N)$	O(M)
E.	O(MlogN)	O(M)

```
public int find(Map<String, Integer> m,
                                String[] can) {
  int res = 0;
  for(String k : can)
    if(m.containsKey(k))
      res++;
  return res;
}
```

QUESTION 33

What is output by the code to the right?

A. `4 56` B. `11 5` C. `9 5`

D. `13` E. `6 56`

```
String gar = "56+=78-412**32^";
String[] rs = gar.split("\\D");
System.out.print(rs.length + " " + rs[0]);
```

QUESTION 34

The following values are inserted in the order shown into a binary search tree using the traditional, naive insertion algorithm. What is the result of post order traversal of the resulting tree?

11, 3, 7, 8, 3, 6, 9, 11, 12

A. 12 11 9 8 7 6 3 B. 3 6 7 8 9 11 12 C. 11 3 7 6 8 9 11 12

D. 6 9 8 7 3 12 11 E. 11 3 7 6 8 9 12

QUESTION 35

What kind of graph does the picture to the right represent?

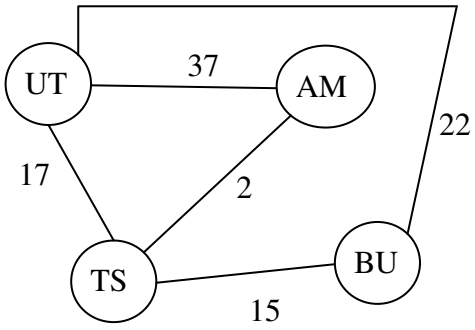
A. a directed unweighted graph

B. a directed weighted graph

C. an undirected unweighted graph

D. a undirected weighted graph

E. a binary search tree



QUESTION 36

What is output by the code to the right?

A. `truetruetrue`

B. `falsefalsetrue`

C. `falsefalsefalse`

D. `truefalsetrue`

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

```
String[] gds = {"A", "", null, "C", "D"};
System.out.print(gds instanceof Object);
System.out.print(gds[2] instanceof Object);
System.out.print(gds[1] instanceof String);
```

QUESTION 37

Given method `process` to the right, what is output by the following client code?

```
int[] p1 = process(9);
System.out.print(p1[4]);
```

A. 90 B. 9 C. 3

D. 2 E. 1

QUESTION 38

Given method `process` to the right, what is output by the following client code?

```
int[] p2 = process(15);
int tot = 0;
for(int i : p2)
  tot += i;
System.out.print(tot);
```

A. 105 B. 36 C. 33

D. 27 E. 14

```
public int[] process(int max) {
  int[] cs = {1, 3, 5};
  int[] ms = new int[max];
  Arrays.fill(ms, max * 10);
  ms[0] = 0;
  for(int i = 1; i < ms.length; i++)
    for(int j = 0; j < cs.length; j++)
      if(cs[j] <= i) {
        int t = ms[i - cs[j]] + 1;
        if(t < ms[i])
          ms[i] = t;
      }
  return ms;
}
```

QUESTION 39

What is output by the client code to the right?

A. 36912151821

B. 3612

C. 369121518

D. 181512963

E. 36120000000

QUESTION 40

What type of data structure does the `Structure` class implement?

A. An array based list

B. A stack

C. A set

D. A min heap

E. A queue

```
public class Structure<E> {
  private ArrayList<E> con;

  public Structure() {
    con = new ArrayList<E>(10);
  }

  public void add(E v) { con.add(v); }

  public E remove() {
    return con.remove(0);
  }

  public E peek() { return con.get(0); }

  public boolean isEmpty() {
    return con.size() == 0;
  }
}

// client code
Structure<Integer> str;
str = new Structure<Integer>();
for(int i = 3; i < 20; i += i)
  str.add(i);
while(!str.isEmpty())
  System.out.print(str.remove());
```

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)

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 District 2 2012

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

Notes:

The clause "Choose the most restrictive correct answer." is necessary because per the formal definition of Big O, an algorithm that is $O(N^2)$ is also $O(N^3)$, $O(N^4)$, and so forth.

31. Negative integers in Java are stored in 2's complement format.

33. When the delimiter is set without the plus sign (`"\\D"` instead of `"\\D+"`) sets of 2 delimiters in a row cause empty `String`s to be created.