University Interscholastic League

Computer Science Competition

Number 135 (Regional - 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 does $F9E_{16}$ minus 110011110001_2 equal?

A. $34D_{16}$ B. $1D8F_{16}$ C. 109_{10} D. $2AD_{16}$ E. $24F_{16}$

QUESTION 2

What is output by the code to the right?

A. 8 B. 108 C. 195

D. 895 E. 995

```
int x = 895;
int y = 100;
int z = x % y + y % x;
System.out.print(z);
```

QUESTION 3

What is output by the code to the right?

A. 16 B. 21 C. 23

D. 26 E. 28

```
int val = -5;
for(int i = val; i <= 8; i++){
  val++;
  ++val;
}
System.out.print(val);
```

QUESTION 4

What is output by the code to the right?

A. 20 B. 12 C. 9

D. 8 E. 6

```
String c1 = "#Yoo*";
String c2 = c1.toLowerCase();
c2 += c1 + c2 + c1;
System.out.print(c2.length());
```

QUESTION 5

What is output by the code to the right?

A. null5 B. 5 C. 4

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

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

```
String[] st = new String[5];
System.out.print(st[3] + st.length);
```

QUESTION 6

What is output by the code to the right?

A. 2.0 B. 2.125 C. 5.125

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

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

```
double a1 = 52.125;
a1 %= 10;
System.out.print(a1);
```

QUESTION 7

Which answer is logically equivalent to the following `boolean` expression, where `p`, `q`, and `r` are `int` variables?

```
!((p < q) || !(q >= r))
```

A. `(p < q) || (q >= r)` B. `(p >= q) && (q >=r)` C. `!(p < q) && !(q >= r)`

D. `p <= r` E. `!(p != q) && (q < r)`

QUESTION 8

What is output by the code to the right?

A. `12` B. `23`

C. `24` D. `14`

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

```
int x1 = 7;
if(x1 * 2 > 10)
  System.out.print(1);
else
  System.out.print(2);
if(x1 == 14)
  System.out.print(3);
else
  System.out.print(4);
```

QUESTION 9

What is output by statement marked `// line 1` in the client code to the right?

A. `3 1503`

B. `1500 4503`

C. `0 0`

D. `3 1500`

E. `3 1800`

QUESTION 10

What is output by statement marked `// line 2` in the client code to the right?

A. `2 300`

B. `2 400`

C. `2 600`

D. `2 800`

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

```
public class School {
  private int numStudents, cls;

  public School(int ns, int c) {
    numStudents = ns;
    cls = c;
  }

  public void newYear() {
    numStudents += cls * 100;
  }

  public String toString() {
    return cls + " " + numStudents;
  }
}

public class BigSchool extends School {
  public BigSchool(int ns, int c) {
    super(ns * 2, c);
  }

  public void newYear() {
    super.newYear();
    super.newYear();
  }
}

// client code
School sc1 = new School(1500, 3);
sc1.newYear();
System.out.print(sc1); // line 1
School sc2 = new BigSchool(100, 2);
sc2.newYear();
System.out.print(sc2); // line 2
```

QUESTION 11

What is output by the code to the right?

A. `54` B. `47` C. `43`

D. `35` E. `1`

```
int m = 35;
int n = 40;
int o = 19;
System.out.print(m & o | n);
```

QUESTION 12

The code to the right contains a syntax error. Which of the following best describes the cause of the syntax error?

A. The `++` operator is not defined for variables of type `double`.

B. The expression `++m1` must be rewritten as `m1++`.

C. Variables of type `double` are not legal arguments for the `Math.round` method.

D. Expressions that result in `double`s may not be assigned to variables of type `int`.

E. The `round` method must be called via an object of type `Math`, not the class name.

```
double m1 = -125;
int n1 = Math.round(++m1);
System.out.print(n1);
```

QUESTION 13

What is output by the code to the right?

A.
```
    "one
two
```

B. `"one"two"`

C. `"onetwo`

D. `\t\onetwo`

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

```
System.out.print("\t\"one");
System.out.println("two");
```

QUESTION 14

What is output by the code to the right?

A. `100000000` B. `000256`

C. `000006` D. `000400`

E. `000000400`

```
int tk = 256;
System.out.printf("%06o", tk);
```

QUESTION 15

What is returned by the method call `change(7471020)`?

A. `0` B. `2` C. `21`

D. `747102` E. `247123`

```
public int change(int x) {
  if(x <= 0)
    return x;
  else
    return change(x / 10) + x % 10;
}
```

QUESTION 16

What is output by the code to the right?

A. `36` B. `37` C. `55`

D. `63` E. `64`

```
String stars = "*";
for(int i = 3; i < 12; i++)
  for(int j = 0; j < i; j++)
    stars += "*";
System.out.print(stars.length());
```

QUESTION 17

What is output by the code to the right?

A. `[0, 1, 2]` B. `[0.0, 1.0, 2.0]`

C. `[0, 0, 0]` D. `3 [0, 1, 2]`

E. The output will vary from one run of the program to the next.

```
int[] values = {0, 1, 2};
System.out.print(values);
```

QUESTION 18

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

A. `0` B. `1` C. `7`

D. `8` E. `15`

```
boolean p, q, r, s;

// code to initialize p, q, r, and s

boolean t = !p && q && r && !s;
```

QUESTION 19

What is output by statement marked `// line 1` in the client code to the right?

A. `0` B. `13` C. `15`

D. `18` E. `26`

QUESTION 20

What is output by statement marked `// line 2` in the client code to the right?

A. `0` B. `15` C. `18`

D. `26` E. `48`

```
public int tinker(int[] data) {
  data[0] += data[2];
  data[1] -= data[1] * 3;
  int t = 0;
  for(int i : data)
    t += i;
  data = new int[4];
  data[0] = 15;
  return t;
}

// client code
int[] readings = {12, 5, 7, 2};
int ans = tinker(readings);
System.out.print(ans); // line 1
ans = 0;
for(int i : readings)
  ans += i;
System.out.print(ans); // line 2
```

QUESTION 21

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

I. `byte`
II. `short`
III. `float`

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

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

```
double value = 48.125627;
<*1> var = (<*1>) value;
```

QUESTION 22

Which of the following can replace **<*1>** in method `work` so that the method compiles without error?

A. `new` B. `work` C. `true`

D. `finally` E. `22_$`

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

```
public int work(int <*1>, int y) {
  int z = <*1> + y;
  <*1>++;
  y++;
  System.out.print(z + " ");
  return z + <*1> + y;
}

// client code
int x = 4;
int y = -5;
System.out.print(x++ + " "
               + work(x++, y) + " " + --y);
```

QUESTION 23

What is output by the client code to the right?

A. `1 5 3 -6` B. `0 4 2 -6`

C. `5 1 3 -5` D. `1 5 3 -4`

E. `4 0 2 -4`

QUESTION 24

What is output by the code to the right?

A. `2false`

B. `1false`

C. `1true`

D. The output will vary from one run of the program to the next.

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

```
boolean p = true;
if(p = false)
  System.out.print(1);
else
  System.out.print(2);
System.out.print(p);
```

QUESTION 25

What is the largest possible value the code to the right will output?

A. `990` B. `999` C. `1000`

D. `1089` E. `1100`

```
int tot = 0;
int lim = ((int) (Math.random() * 10)) + 1;
for(int i = 0; i < lim; i++) {
  int temp = (int) (Math.random() * 100);
  tot += temp;
}
System.out.print(tot);
```

QUESTION 26

Which of the following can replace **<*1>** in the code to the right so that the output is `6`?

A. `Character.isLowerCase(ch)`

B. `!Character.isLowerCase(ch)`

C. `Character.isLetter(ch)`

D. `!Character.isLetter(ch)`

E. `Character.isLetterOrDigit(ch)`

```
String uni = "Texas-Tech-2011";
int total = 0;
for(int i = 0; i < uni.length(); i++) {
  char ch = uni.charAt(i);
  if( <*1> )
    total++;
}
System.out.print(total);
```

QUESTION 27

What replaces **<*1>** in the code to the right so that when the `while` loop is complete `stck.size()` returns `0`?

A. `stck.pop()` B. `stck.isEmpty()`

C. `!stck` D. `!stck == 0`

E. `!stck.isEmpty()`

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

```
Stack<Integer> stck = new Stack<Integer>();
stck.push(-5);
stck.push(10);
if(stck.peek() > 0)
  stck.push(stck.peek());
while( <*1> )
  System.out.print(stck.pop());
```

QUESTION 28

What is output by the code to the right?

A. `1010-5` B. `-510`

C. `-51010` D. `10-5`

E. `10-5-5`

QUESTION 29

What is output by the statement to the right marked `// line 1`?

A. `false` B. `true` C. `-50`

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

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

```
Comparable c1 = "Baylor";
Comparable c2 = "tcu";
boolean b3 = c1.compareTo(c2) > 0;

System.out.print(b3); // line 1

System.out.print(c1.equals(c2)); // line 2
```

QUESTION 30

What is output by the statement to the right marked `// line 2`?

A. `false` B. `true` C. `50`

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

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

QUESTION 31

What is output by the code to the right?

A. `false false`

B. `false true`

C. `true false`

D. `true true`

E. The output will vary from one run of the program to the next.

```
TreeMap<Integer, String> tm;
tm = new TreeMap<Integer, String>();
tm.put(0, "A");
tm.put(12, "B");
tm.put(0, "C");
HashMap<Integer, String> hm;
hm = new HashMap<Integer, String>();
hm.put(0, "C");
hm.put(12, "" + 'B');
System.out.print(tm instanceof Collection);
System.out.print( " " + tm.equals(hm));
```

QUESTION 32

What is returned by the method call `test(7)`?

A. `-2` B. `19` C. `20`

D. `30` E. `38`

```
public int test(int x) {
  if(x <= 2)
    return x * 2;
  return x * 2 + test(x - 2) + test(x - 4);
}
```

QUESTION 33

What is the worst case order (Big O) of method `slide` to the right? N = `d1.length` and M = `d2.length`. Pick the most restrictive correct answer.

A. O(NM) B. O(NlogM)

C. O(N) D. O(MlogN)

E. $O(N^2)$

```
public int slide(int[] d1, int[] d2) {
  int res = 0;
  for(int i = 0; i < d1.length; i++)
    for(int j = i; j < i + 10; j++)
      if(j >= d2.length)
        break;
      else if(d2[j] > d1[i])
        res += d2[j];
  return res;
}
```

QUESTION 34

What is output by method `sort` when the following client code is executed?

```
int[] tst = {37, 52, 16, 8, 21, 53};
sort(tst);
```

A. `[21, 16, 8][53, 52, 37]`

B. `[16, 8][37, 21, 53, 52]`

C. `[37, 21, 53, 52, 16][8]`

D. `[8, 37][21, 53, 52, 16]`

E. `[53, 52, 37, 21, 16, 8][]`

QUESTION 35

What sorting algorithm does method `sort` implement?

A. selection sort

B. insertion sort

C. quicksort

D. merge sort

E. radix sort

```
// pre: all values in data > 0
public void sort(int[] data) {
  ArrayList<Integer>[] t =
   (ArrayList<Integer>[]) new ArrayList[2];
  t[0] = new ArrayList<Integer>();
  t[1] = new ArrayList<Integer>();
  int b = 1;
  for(int i = 0; i < 31; i++) {
    for(int j = 0; j < data.length; j++)
      t[(data[j] & b) / b].add(data[j]);
    b = b << 1;
    int j = 0;
    for(int x : t[1])
      data[j++] = x;
    for(int x : t[0])
      data[j++] = x;
    if(i == 4)
      System.out.print(t[0] + "" + t[1]);
    t[0].clear();
    t[1].clear();
  }
}
```

QUESTION 36

Assume method `regional(int[] data)` is $O(2^N)$ where N = `data.length`. When method `regional` is passed an array with `length` = 200 it takes 3 seconds for method `regional` to complete. If method `regional` is then passed an array with `length` = 207 what is the expected time it will take method `regional` to complete?

A. 3.1 seconds B. 128 seconds C. 384 seconds D. 1,024 seconds E. 3,072 seconds

QUESTION 37

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

```
int[] fs = {0, 8, 4, 12, 2, 10, 6, 14,
                          1, 9, 5, 13};

for(int i : prc(fs))
  System.out.print(i + " ");
```

A. `1 2 1 2 1 2 1 2 1 2 1 2`

B. `1 2 2 3 2 3 3 4 3 4 4 6`

C. `0 8 4 12 2 10 6 14 1 9 5 13`

D. `1 2 2 3 2 3 3 4 2 4 3 5`

E. `1 2 2 4 2 4 4 8 2 6 5 11`

```
public int[] prc(int[] data) {
  int[] f = new int[data.length];
  for(int i = 0; i < data.length; i++) {
    int m = 0;
    for(int j = 0; j < i; j++)
      if(data[i] > data[j] && f[j] > m)
        m = f[j];
    f[i] = m + 1;
  }
  return f;
}
```

QUESTION 38

Method `wrong` to the right has a syntax error. Which of the following best describes the syntax error?

A. `_` is not a valid identifier.

B. `C` is not a valid identifier for a variable.

C. The expression `C = 4` must be changed to `C == 4`.

D. The parameter `t` may not be declared to be `final`.

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

```
public void wrong(int[] _, final int t) {
  int C = 0;
  for(int wrong : _) {
    if(_[wrong] == t) {
      _[wrong]++;
      C++;
    }
    if(C = 4)
      return;
  }
}
```

GO ON TO THE NEXT PAGE.

QUESTION 39

What is output by the following client code?

```
Structure st = new Structure();
st.add(0, "A");
st.add(0, 12);
st.add(1, 0.5);
st.add(st.size(), "B");
for(int i = 0; i < st.size(); i++)
  System.out.print(st.get(i) + " ");
```

A. `12 0.5 B`

B. `A 0.5 12 B`

C. `12 0.5 A B`

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

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

QUESTION 40

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

A. An array based list

B. A linked list

C. A stack

D. A queue

E. A graph

```
public class Structure<E> {
  private N<E> st = new N<E>(null, null);
  private int s;

  public void add(int i, E v) {
    N<E> n = new N<E>(v, g(i));
    g(i - 1).n = n;
    s++;
  }

  public E get(int i) { return g(i).d; }

  public void remove(int i) {
    g(i - 1).n = g(i).n;
    s--;
  }

  public int size() { return s; }

  private N<E> g(int i) {
    N<E> t = st;
    for(int j = -1; j < i; j++, t = t.n);
    return t;
  }

  private static class N<E> {
    private E d;
    private N<E> n;

    private N(E d1, N<E> n1) {
      d = d1;
      n = n1;
    }
  }
}
```

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 Regional 2012

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

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.

17. The hashcode of the variable is printed, which will vary from one run of the program to the next. Printing the contents of the array requires a call to `Arrays.toString(values)` or a loop to manually print each element.

31. `Map`s do not implement the `Collection` interface. `TreeMap`s and `HashMap`s are equal if they contain the same key-value pairs even though they may store them in different orders.