University Interscholastic League

Computer Science Competition

Number 132 (Invitational B - 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 101_2 times 111_2 equal?

A. -10_2 B. 1100_2 C. 101111_2 D. 111101_2 E. 100011_2

QUESTION 2

What is output by the code to the right?

A. 18 B. 93 C. 793

D. 869 E. 1017

```
int x = 1776;
int y = x % 1000 + x / 100;
System.out.print(y);
```

QUESTION 3

What is output by the code to the right?

A. 2 B. 14 C. 24

D. 30 E. 222222222222

```
int val = 0;
for(int i = -2; i <= 12; i++)
  val += 2;
System.out.print(val);
```

QUESTION 4

What is output by the code to the right?

A. rmian.Basi B. ermian.Bas

C. rmian D. rmian.

E. UT.Permian.Basin

```
String c1 = "UT.Permian.Basin";
String c2 = c1.substring(5, 10);
System.out.print(c2);
```

QUESTION 5

What is output by the code to the right?

A. 7 1 B. 6 1 C. 7 13

D. 7 4 E. 6 13

```
int[] st = {5, 3, 13, 4, -1, 6, 0};
System.out.print(st.length + " " + st[3]);
```

QUESTION 6

What is output by the code to the right?

A. 9 B. 10 C. 11

D. 12 E. 20

```
int x1 = 3;
int y1 = 2;
int z1 = x1++ * ++y1;
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. 7 B. 5 C. 4

D. 1 E. 0

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

boolean s = !p || !q || !r;
```

QUESTION 8

What is output by the code to the right?

A. 11 B. 12 C. 1x2

D. 15 E. 25

```
int x2 = 5;
if(x2 % 2 == 0)
  System.out.print(1);
else
  System.out.print(2);
System.out.print(x2);
```

QUESTION 9

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

A. 321

B. 123

C. 3

D. 1

E. 31

QUESTION 10

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

A. numStudentsisPrivate

B. "101"

C. 101

D. true100

E. 100true

```
public class School {
  private boolean isPrivate;
  private int numStudents;

  public School() {
    this(true);
    System.out.print(1);
  }

  public School(boolean p) {
    this(100, p);
    System.out.print(2);
  }

  public School(int ns, boolean p) {
    isPrivate = p;
    numStudents = ns;
    System.out.print(3);
  }

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

// client code
School sc = new School(); // line 1
System.out.print(sc); // line 2
```

QUESTION 11

What is output by the code to the right?

A. 1 B. 3 C. 10

D. 15 E. 21

```
int m = 0xA;
int n = 31;
System.out.print(m ^ n);
```

QUESTION 12

What is output by the code to the right?

A. 15.0 B. 6.0 C. 5.5

D. 4.0 E. 0

```
double m1 = 30.0;
m1 = Math.max(Math.sqrt(m1), m1 / 2);
System.out.print(m1);
```

QUESTION 13

What is output by the code to the right?

A.
```
red\
blue\
pink\
```

B.
```
red
blue
pink
```

C.
```
red
bluepink
```

D.
```
redblue
pink
```

E.
```
redbluepink
```

```
System.out.print("red\nblue\npink");
```

QUESTION 14

What is output by the code to the right?

A. `6.0`

B. `6528221.0`

C. `652.1`

D. `6,528,221.00`

E. `6,528,221.0`

```
double mon = 6528221.00;
System.out.printf("%,3.1f", mon);
```

QUESTION 15

What is returned by the method call `b(5)`?

A. `21`

B. `24`

C. `30`

D. `36`

E. `42`

```
public int a(int x, int z) {
  x++;
  z *= 2;
  return x + z;
}

public int b(int y) {
   return y + a(y, y);
}
```

QUESTION 16

What is output by the code to the right?

A. `50`

B. `30`

C. `25`

D. `15`

E. `10`

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

QUESTION 17

Method `check` to the right will not compile due to a syntax error. Which of the following best describes the syntax error(s) in method `check`?

A. The line `x += a % 10;` causes a loss of precision error.

B. `&` is not a valid `boolean` operator.

C. Variables may not be named `continue`.

D. `x < 1000.00` is not a valid `boolean` expression.

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

```
public int check(double a) {
  boolean continue = true;
  int x = 0;
  while(continue & a > 1.0) {
    x += a % 10;
    a /= 10;
    continue = x < 1000.00;
  }
  return x;
}
```

QUESTION 18

What is the smallest possible value that will be printed out by the code to the right?

A. `0` B. `10` C. `30`

D. `60` E. `70`

```
int total = 0;
for(int i = 0; i < 10; i++)
  total += (int)(Math.random() * 4) + 3;
System.out.print(total);
```

QUESTION 19

Which of the following can replace **<*1>** in the code to the right so that the code segment compiles with error?

A. `double` B. `float`

C. `int` D. `long`

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

```
int xVal = 45;
int yVal = 100 * xVal;
<*1> vel = xVal / yVal;
```

QUESTION 20

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

```
p ^ q
```

A. `(!p && !q)` B. `(p && !q) || (!p && q)` C. `(!p || !q)`

D. `(!p || p) && (!q || q)` E. `(p && q) && !(p || q)`

QUESTION 21

What replaces **<*1>** in the code to the right to handle all values of `gm` that are not explicitly handled by one of the `case` sections?

A. `goto` B. `default` C. `case`

D. `break` E. `switch`

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

QUESTION 22

What is returned by the method call `pts("WWSLTLDSS")`?

A. `1013` B. `48` C. `45`

D. `17` E. `15`

```
public int pts(String res) {
  int t = 0;
  for(int i = 0; i < res.length(); i++) {
    char gm = res.charAt(i);
    switch(gm) {
      case 'D': t += 1; break;
      case 'L': t -= 1; break;
      case 'S': t += 4; break;
      case 'T': t += 2; break;
      case 'W': t += 1; break;
      <*1> : t += 1000; break;
    }
  }
  return t;
}
```

QUESTION 23

What is output by the code to the right?

A. `null` B. `10`

C. `1` D. `0`

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

```
ArrayList<String> names;
names = new ArrayList<String>();
System.out.print(names.size());
```

QUESTION 24

What is output by the code to the right?

A. 24 B. 21 C. 15

D. 0 E. -5

```
int[] scs = {-5, 5, 2, -2, -5, 5};
int temp = 0;
for(int i : scs)
  temp = i + temp;
System.out.print(temp);
```

QUESTION 25

Given an array of 1000 elements in sorted order what is the largest possible value that will be printed when the array is passed to method `mystery`?

A. 0 B. 1 C. 9

D. 10 E. 500

QUESTION 26

Which algorithm does method `mystery` implement?

A. insertion sort B. selection sort

C. linear search D. radix sort

E. binary search

```
public int mystery(int[] v, int t) {
  int w = 0;
  int h = v.length - 1;
  int c = 0;
  while(w <= h) {
    c++;
    int m = (w + h) >>> 1;
    if(v[m] < t) h = m - 1;
    else if(v[m] > t) w = m + 1;
    else {
      System.out.print(c);
      return m;
    }
  }
  System.out.print(c);
  return -(w + 1);
}
```

QUESTION 27

Consider the following timing data for method `sort` shown to the right and various arrays:

array W: 1,000,000 elements in random order. Method `sort` takes 10 second to complete.

array X: 1,000,000 elements in ascending order. Method `sort` takes 100 seconds to complete.

What is the expected time for method `sort` to complete given array Y with 2,000,000 elements in random order and array Z with 2,000,000 elements in ascending order?

	array Y	array Z
A.	10 seconds	100 seconds
B.	11 seconds	400 seconds
C.	20 seconds	200 seconds
D.	21 seconds	210 seconds
E.	21 seconds	400 seconds

QUESTION 28

Which sorting algorithm do methods `hp` and `sort` implement?

A. radix sort B. mergesort

C. heap sort D. quicksort

E. selection sort

```
public void hp(double[] v, int i, int j) {
  double t = v[i];
  v[i] = v[j];
  v[j] = t;
}

public void sort(double[] v, int s, int p){
  if(s < p) {
    int m = (s + p) / 2;
    hp(v, m, s);
    int i, j = s;
    for(i = s + 1; i <= p; i++ )
      if(v[i] <= v[s]) {
        j++;
        hp(v, i, j);
      }
    hp(v, s, j);
    sort(v, s, j - 1);
    sort(v, j + 1, p);
  }
}
```

QUESTION 29

Which of the following replaces **<*1>** in the code to the right to indicate the `TDPoint` class is a subclass of the `Point` class?

A. `final` B. `static` C. `extends`

D. `super` E. `implements`

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

QUESTION 30

What is output by the following client code?

```
Point p1 = new Point(5, 2);
Point p2 = new Point(5, 2);
System.out.print(p1 == p2);
System.out.print(" " + p1.equals(p2));
```

A. `false false`

B. `false true`

C. `true false`

D. `true true`

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

QUESTION 31

What is output by the following client code?

```
TDPoint p3 = new TDPoint(2, 3, 4);
System.out.print(p3);
```

A. `2324` B. `235`

C. `11` D. `236`

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

```
public class Point {
  private int x, y;

  public Point(int xn, int yn) {
    x = xn;
    y = yn;
  }

  public int dFact() { return x * y; }

  public String toString() {
    return "" + x + y + dFact();
  }
}

public class TDPoint <*1> Point {
  private int z;

  public TDPoint(int xn, int yn, int zn) {
    super(xn, yn);
    z = zn;
  }

  public int dFact() {
    return super.dFact() * z;
  }
}
```

QUESTION 32

What is returned by the method call `tester(20)`?

A. `80` B. `40` C. `10`

D. `5` E. `2`

```
public int tester(int x) {
  try {
    if(x < 10) return x * 2;
    return 100 / x;
  }
  finally { x *= 2; }
}
```

QUESTION 33

What is output by the code to the right?

A. `-5.15` B. `0.0` C. `0.15`

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

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

```
PriorityQueue<Double> pq;
pq = new PriorityQueue<Double>();
pq.add(0.15);
pq.add(-5.15);
pq.add(0.0);
System.out.print(pq.peek());
```

QUESTION 34

Given the undirected, weighted graph to the right, what is the cost of the lowest cost path from vertex A to vertex E?

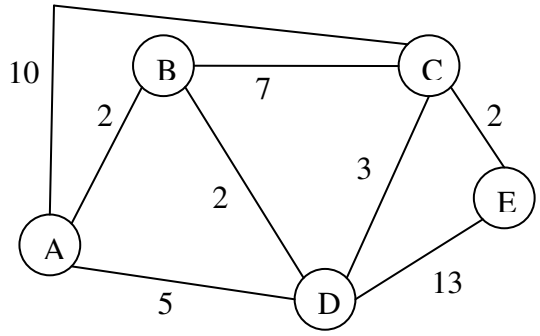
A. 1

B. 2

C. 9

D. 12

E. 14



QUESTION 35

What is output by the code to the right?

A. 0

B. 1

C. 25

D. 49

E. 81

```
int[][][] cb = {{{5, 6, 2}, {4, 3, 1}},
                {{2, 1, 4}, {7, -6, 1}}};
int m = 0;
for(int i = 0; i < cb[0].length; i++)
  for(int j = 0; j < cb[0][0].length; j++){
    int t = cb[0][i][j] - cb[1][i][j];
    t *= t;
    if(t > m)
      m = t;
  }
System.out.print(m);
```

QUESTION 36

What is output by the code to the right?

A. 0 B. 1 C. 4

D. 5 E. 6

```
String[] cs = {"ut", "ttu", "tamu", "tsu",
                                   "tu", "nt"};
int ct = 0;
for(String str : cs)
  if(str.matches("t.+"))
    ct++;
System.out.print(ct);
```

QUESTION 37

What is output by the client code to the right?

A. falsefalse

B. falsetrue

C. truefalse

D. truetrue

E. false is output and then a runtime error occurs.

```
public boolean check(int s, int b, int g) {
  return g <= b * 5 + s && g % 5 <= s;
}

// client code
System.out.print(check(3, 2, 8));
System.out.print(check(6, 0, 11));
```

GO ON TO THE NEXT PAGE.

QUESTION 38

What replaces **<*1>** in the `access` and `remove` methods to the right so that the methods generate an exception if the `boolean` expression `d == t` is `true`?

A. `throw` B. `try` C. `catch`

D. `double` E. `throws`

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

QUESTION 39

What is output by the following client code?

```
Structure gar = new Structure();
gar = gar.add("LHN");
gar = gar.add(24);
gar = gar.add('A');
while(!gar.isEmpty()) {
  System.out.print(gar.access());
  gar = gar.remove();
}
```

A. `LHN`

B. `LHNA24`

C. `A`

D. `A24LHN`

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

QUESTION 40

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

A. a set

B. a queue

C. a binary search tree

D. a min heap

E. a stack

```
public class Structure {

  private static final Object t;
  static { t = new Object(); }

  private Object d;
  private Structure n;

  public Structure() { d = t; }

  public Structure add(Object d) {
    Structure r = new Structure();
    r.d = d;
    r.n = this;
    return r;
  }

  public Object access() {
    if(d == t)
      <*1> new IllegalStateException();
    return d;
  }

  public boolean isEmpty() {return d == t;}

  public Structure remove() {
    if(d == t)
      <*1> new IllegalStateException();
    return n;
  }
}
```

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)

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)