University Interscholastic League

Computer Science Competition

Number 116 (District 2 - 2009)

General Directions (Please read carefully!):

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 10001_2 and 1011_2?

A. 11100_2 B. 11111_2 C. 10111_2 D. 11000_2 E. 11110_2

QUESTION 2

What is output by the code to the right?

A. 0 B. 0.4 C. 40

D. 2 E. 2.5

```
int x = 10;
int y = 4;
x = y / x;
System.out.println( x );
```

QUESTION 3

What is output by the code to the right?

A. 7 B. 18 C. 14

D. 6 E. 12

```
int accum = 0;
for(int i = 0; i <= 6; i++){
  accum++;
  accum++;
}
System.out.print( accum );
```

QUESTION 4

What is output by the code to the right?

A. 1 B. 0 C. -1

D. 4 E. 3

```
String prog = "haskell";
System.out.print( prog.indexOf('E', 3) );
```

QUESTION 5

What is output by the code to the right?

A. 2 B. 7 C. 5

D. 8 E. 10

```
int[] data = {2, 3, 1, 5, 3, 1};
data[1] += data[2] + data[5];
System.out.println( data[1] );
```

QUESTION 6

What is output by the code to the right?

A. 100 B. 35 C. 15

D. 50 E. 12

```
int r = 5;
int s = 2;
r *= s + r;
System.out.print( r );
```

QUESTION 7

What is output by the code to the right?

A. true true

B. true false

C. false true

D. false false

E. false true false true

```
boolean p = (4 > 5);
boolean q = (0 != 0);
System.out.print( p && q );
System.out.print( " " );
System.out.print( !p || p );
```

QUESTION 8

What is output by the code to the right?

A. 8 8 B. 8 9 C. 7 7

D. 7 9 E. 7 8

```
int x = 8;
int y = 4 * 2;
if( x == y ){
  if( x % 2 == 0 )
    x--;
  else
    y++;
}
System.out.print( x + " " + y );
```

QUESTION 9

What replaces **<*1>** in the code to the right so that `MAX_PTS` and `PASS_RATE` are class constants that are accessible only in the `Grade` class?

A. `public final`

B. `private static`

C. `private final`

D. `private void final`

E. `private static final`

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

```
public class Grade{

  <*1> int MAX_PTS = 100;
  <*1> double PASS_RATE = 0.7;

  private int points;

  public Grade(int p){
    points = p;
  }

  public boolean pass(){
    double ave = 1.0 * points / MAX_PTS;
    return ave >= PASS_RATE;
  }
}
```

QUESTION 10

What is output by the client code to the right?

A. `true` B. `false` C. `1`

D. `180` E. `true true`

```
////////////////////////////////////////
// client code
Grade hist = new Grade(75);
Grade cs = new Grade(105);
boolean result = hist.pass() && cs.pass();
System.out.print( result );
```

QUESTION 11

What is output by the code to the right?

A. 3 B. 15 C. 77

D. 4 E. 12

```
int m = 11;
int n = 7;
System.out.print( m ^ n );
```

QUESTION 12

What is output by the code to the right?

A. 0 B. 24 C. 12

D. 36 E. -12

```
int x = 12;
System.out.print( Math.abs(x) + x );
```

QUESTION 13

What is output by the code to the right?

A. `ipipip` B. `ipip` C. `ip`

D. `yyyyip` E. `ipipipip`

```
String text = "ip";
System.out.print( text + text );
System.out.print( text );
```

QUESTION 14

What is output by the code to the right?

A. `03.10` B. `3.10000` C. `0x3.10`

D. `3.10x8` E. `003.10`

```
System.out.printf("%05.2f", 3.1);
```

QUESTION 15

What is returned by the method call `joy(-3)`?

A. `11` B. `-1` C. `-3`

D. `5` E. `-13`

```
public int joy(int w){
  w = w * w;
  w -= w;
  w--;
  return w;
}
```

QUESTION 16

What is output by the code to the right?

A. `a` B. `99` C. `c`

D. `2` E. `y`

```
int let = 'a';
let += 2;
System.out.print( (char)let );
```

QUESTION 17

What is output by the code to the right?

A. `3mid3` B. `3mid12` C. `3sum3`

D. `12mid12` E. `12mid3`

```
String sum = 1 + 2 + "mid" + 1 + 2;
System.out.print( sum );
```

QUESTION 18

What is output by the code to the right?

A. `[2, 1]` B. `[1, 2]` C. `[2, 0]`

D. `[0, 2]` E. `[2, 0, 1]`

```
ArrayList<Integer> readings;
readings = new ArrayList<Integer>();
readings.add(2);
readings.add(0, 1);
System.out.print( readings );
```

QUESTION 19

What is output by the code to the right?

A. `3` B. `12` C. `4`

D. `6` E. There is no output due to an infinite loop.

```
int test = 3;
int flag = 6;
do{
  test++;
  flag *= 2;
} while( flag < test );
System.out.print( test );
```

QUESTION 20

What is output by the code to the right?

A. `0` B. `null` C. `1`

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

E. There is no output due to a `NullPointerException`.

```
String[] courses = new String[5];
System.out.print( courses[4].length() );
```

QUESTION 21

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

I. `name`
II. `(Object)name`
III. `name.toObject()`

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

D. I and II E. II and III

```
String name = "Sam";
Object obj;
obj = <*1>;
System.out.print( obj );
```

QUESTION 22

What is output by the code to the right?

A. Abe B. P C. ?:

D. V E. Mondale

```
String pres = "Abe";
String vice = "Mondale";
char res;
res = ( pres.length() > vice.length() )
          ? 'P' : 'V';
System.out.print( res );
```

QUESTION 23

What is output by the code to the right?

A. 3 B. 2 C. 13

D. 12 E. 10

```
int hold = 10;
int other = 2;
if((hold % 5 == 0) || (other++ % 2 == 0))
  hold += other;
System.out.print( hold );
```

QUESTION 24

What replaces **<*1>** in the code to the right to indicate the block of code that sets `count` to `-1` is the exception handling code for any `IOExceptions` generated by the code in the preceding `try` block?

A. `catch(IOException e)`
B. `finally(IOException e)`
C. `then`
D. `catch`
E. `throws(RuntimeException e)`

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

QUESTION 25

If no file exists with the name specified by the `String nm` what does method `count` do?

A. Returns `-1`.
B. Returns `0`.
C. Returns `null`.
D. The program halts due to a runtime error.
E. Method `count` never ends due to an infinite loop.

```
public int count(String nm){
  int count = 0;
  try{
    FileReader f;
    f = new FileReader( new File(nm) );
    while( f.ready() ){
      f.read();
      count++;
    }
  } <*1> {
    count = -1;
  }
  return count;
}
```

QUESTION 26

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

I. `26.2`
II. `new Double(26.2)`
III. `"26.2"`

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

D. I and II E. I and III

```
ArrayList<Double> distances;
distances = new ArrayList<Double>();
distances.add(<*1>);
```

QUESTION 27

What is output by the client code to the right?

A. `[5, 1, -5, 0, 2]`

B. `[5, 2, 1, 0, -5]`

C. `[0, 1, 2, 3, 4]`

D. `[5, 2, 1, 0]`

E. `[-5, 0, 1, 2, 5]`

QUESTION 28

Which searching algorithm does method `find` use?

A. sequential search

B. interpolation search

C. quick search

D. linear search

E. binary search

QUESTION 29

Which sorting algorithm does method `sort` implement?

A. a modified insertion sort

B. a modified radix sort

C. a modified selection sort

D. a modified quick sort

E. a modified merge sort

```
/* If tgt is present in nums return the
index of an element equal to tgt
else return the index of where tgt should
be placed to maintain nums in sorted order.
*/
public int find(int[] nums,
                          int tgt, int high){
  int low = 0;
  int mid = (low + high) / 2;
  boolean found = false;
  while( !found && low <= high){
    mid = (low + high) / 2;
    if( nums[mid] < tgt )
      low = mid + 1;
    else if( nums[mid] > tgt )
      high = mid - 1;
    else
      found = true;
  }
  return found ? mid : low;
}

public void sort(int[] nums){
  for(int i = 1; i < nums.length; i++){
    int tgt = find(nums, nums[i], i - 1);
    int temp = nums[i];
    for(int j = i; j > tgt; j--){
      nums[j] = nums[j -1];
    }
    nums[tgt] = temp;
  }
}

// client code
int[] nums = {5, 1, -5, 0, 2};
sort(nums);
System.out.print( Arrays.toString(nums) );
```

QUESTION 30

What is the Big O of method `range`? The `LinkedList data` contains N distinct `Intgers`. Pick the most restrictive correct answer.

A. O(NlogN) B. $O(N^{3/2})$ C. $O(N^2 logN)$

D. $O(N^2)$ E. O(N)

```
public int range(LinkedList<Integer> data){
  Collections.sort( data );
  int min = data.getFirst();
  int max = data.getLast();
  return max - min + 1;
}
```

QUESTION 31

What is output by the code to the right?

A. 31 B. 0 C. 64

D. 128 E. 223

```
int alpha = 64;
int beta = 31;
int gamma = 128;
gamma = alpha & beta | gamma;
System.out.println( gamma );
```

QUESTION 32

Which of the following replaces **<*1>** in the code to the right to return the value in `max` if the value in `max` is greater than or equal to the number of elements in array `d` from index `i` to the last element in the array, inclusive?

A.
```
if( max >= lim - i )
  return max;
```

B.
```
if( max >= i )
  break;
```

C.
```
if( max >= d.length )
  return max;
```

D.
```
if( max >= d[i] )
  return max;
```

E.
```
if( max > d.length - d[i] )
  break;
```

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

```
public int trace(int[] d){
  int max = 0;
  int lim = d.length;
  for(int i = 0; i < lim; i++){
    <*1>
    int j = i + 1;
    int len = 1;
    while( j < lim && d[j] > d[j - 1] ){
      len++;
      j++;
    }
    max = Math.max(max, len);
  }
  return max;
}

// client code
int[] values = {2, 1, -1, 4, 8, 8, 10};
System.out.print( trace(values) );
```

QUESTION 33

What is output by the client code to the right?

A. 4 B. 7 C. 2

D. 10 E. 3

QUESTION 34

Assume method `sample(int[] data)` is $O(N^3)$ where N = `data.length`. When method `sample` is passed an array with `length` = 2,000 it takes 1 second for method `sample` to complete. If method `sample` is then passed an array with `length` = 4,000 what is the expected time it will take method `sample` to complete?

A. 4 seconds B. 9 seconds C. 64 seconds D. 16 seconds E. 8 seconds

QUESTION 35

What is output by the code to the right?

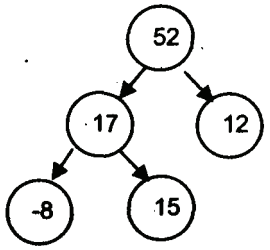
A. `[4, 2]`
B. `[2, 4, 6, 12]`
C. `[6, 12]`
D. `[6, 2, 4, 12]`
E. `[2, 4]`

```
Set<Integer> s1 = new TreeSet<Integer>();
Set<Integer> s2 = new TreeSet<Integer>();
int[] data1 = {6, 2, 4, 12};
int[] data2 = {0, 5, 4, 2};
for(int i = 0; i < data1.length; i++){
  s1.add(data1[i]);
  s2.add(data2[i]);
 }
s1.retainAll(s2);
System.out.print(s1);
```

QUESTION 36

Consider the tree to the right. What kind of tree is it?

A. A stack tree
B. A min heap
C. A max heap
D. A red black tree
E. A binary search tree



QUESTION 37

Consider the code to the right. Which of the following data types can replace **<*1>** in the following client code so that the client code compiles without error?

```
<*1> currentScore = new Score();
```

A. `Score, int`
B. `Object, Incrementable, and Score`
C. `E, Object, Comparable, and Score`
D. `Object, String, and Score`
E. `String, Object, Incrementable, and Score`

```
public interface Incrementable{
  public void increment();
}

public class Score implements
Incrementable{

  private int points;

  public Score(){
    points = 0;
  }

  public void increment(){
    points++;
  }
}
```

QUESTION 38

What replaces **<*1>** in the code to the right to allocate a new array of the proper type with `cap` elements?

A. `new E`

B. `(E[])(new Object[cap])`

C. `E[cap]`

D. `(E)(new Object[])`

E. `new E[cap]`

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

QUESTION 39

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

A. A list

B. A stack

C. A queue

D. A binary search tree

E. A hash table

```
public class Structure<E>{
  private int size;
  private E[] con;

  public Structure(){ con = getCon(10); }

  public void add(E obj){
    if( size == con.length )
      con = getCon( size * 2 );
    con[size++] = obj;
  }

  public E get(int pos){
    return con[pos];
  }

  public void remove(int pos){
    size--;
    for(int i = pos; i < size; i++)
            con[i] = con[i + 1];
   }

  public int size(){ return size; }

  private E[] getCon(int cap){
    E[] temp = <*1>;
    for(int i = 0; i < size; i++)
      temp[i] = con[i];
    return temp;
  }
}
```

QUESTION 40

What is output by the client code to the right?

A. 131 B. 13 C. 1

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

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

```
public interface Card{
  public static final int ACE = 13;
}

public class BlackjackCard implements Card{
  public static final int ACE = 1;
}

// client code
System.out.print( BlackjackCard.ACE );
```

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()

class java.util.ArrayList<E> implements List<E>
Methods in addition to the List methods:
- 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.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
UIL District 2 2009

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

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.