University Interscholastic League

Computer Science Competition

Number 107 (Invitational A - 2008)

General Directions (Please read carefully!):

1) DO NOT OPEN EXAM UNTIL TOLD TO DO SO.

2) **NO CALCULATORS 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 1001_2 plus 1110_2 equal?

A. 10111_2 B. 1001_2 C. 11111_2 D. 111_2 E. 32_{10}

QUESTION 2

What is output by the code to the right?

A. 9 B. 10 C. 12

D. 3 E. xyx

```
int x = 3;
int y = 2;
System.out.println( x + y * x );
```

QUESTION 3

What is output by the code to the right?

A. 10 B. 5 C. 0

D. 6 E. 12

```
int total = 0;
for(int i = 0; i <= 5; i++){
  total += 2;
}
System.out.println( total );
```

QUESTION 4

What is output by the code to the right?

A. SOUTH B. SOUTH88 C. SOUTH**

D. South88 E. SOUTH+**

```
String s = "South88";
System.out.println( s.toUpperCase() );
```

QUESTION 5

What is output by the code to the right?

A. 7 B. 3

C. 2 D. 5

E. 1

```
int[] data = {3, 2, 4, 3, 1, 0};
data[1] = data[1] + data[3];
System.out.println( data[1] );
```

QUESTION 6

What is output by the code to the right?

A. 0 B. 20 C. 0.3

D. 120 E. 6

```
int r = 6;
int v = 20;
System.out.println( r % v );
```

QUESTION 7

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

QUESTION 8

What is output by the code to the right?

A. 21 B. 2 C. 1

D. 12 E. There is no output.

```
double a = 2.5;
double b = 15.7;
if( a < b )
  System.out.print( 1 );
if( b > 10 )
  System.out.print( 2 );
```

QUESTION 9

What replaces **<*1>** in the code to the right to indicate that the method `takeTrip` does not return a value?

A. `return`

B. `null`

C. `static`

D. `private`

E. `void`

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

QUESTION 10

Which of the following creates a `Car` object whose `miles` instance variable is initialized to zero?

A. `Car c = new Car("0");`

B. `Car c = new Car('0');`

C. `Car c = new Car(0);`

D. `Car c = new Car(miles.0);`

E. `Car c = new Car("zero");`

```
public class Car{
  private int miles;

  public Car(int m){
    miles = m;
  }

  public <*1> takeTrip(int len){
    miles += len;
  }

  public int getDistance(){
    return miles;
  }

}
```

QUESTION 11

What is output by the code to the right?

A. 13 B. 11 C. 2

D. 9 E. 0

```
int z = 2;
int k = 11;
System.out.print( k & z );
```

QUESTION 12

How many lines of output does the code to the right produce?

A. 0 B. 1 C. 2

D. 3 E. 4

```
System.out.print("first string");
System.out.print("second string");
System.out.println("third string");
```

QUESTION 13

What is output by the code to the right?

A. 7.0 B. 14 C. 7

D. 14.0 E. 2

```
System.out.println( Math.min(14, 7) );
```

QUESTION 14

What is output by the code to the right?

A. `0019` B. `19.0` C. `000019`

D. `19.00` E. `19`

```
System.out.printf("%04d", 19);
```

QUESTION 15

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

A. `6` B. `3` C. `10`

D. `8` E. `0`

```
public static int simple(int x){
  x++;
  return x + x;
}
```

QUESTION 16

What is output by the code to the right?

A. `2` B. `4` C. `5`

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

E. There is no output due to an `ArrayIndexOutOfBoundsException`.

```
String names = "Bob Don J Tim";
String[] chopped = names.split("\\s+");
System.out.print( chopped.length );
```

QUESTION 17

What is returned by the method call `rec(4)`?

A. `4` B. `1` C. `24`

D. `10` E. `-1`

```
public static int rec(int x){
  if(x <= 1)
    return 1;
  else
    return x + rec(x - 1);
}
```

QUESTION 18

What is output by the code to the right when method `two` is called?

A. `3` B. `4` C. `1`

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

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

```
public static int one(int x){
  return x + x;
}

public static int one(int x, int y){
  return x + y;
}

public static void two(){
  System.out.print( one(2, 1) );
}
```

QUESTION 19

What is output by the code to the right?

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

C. `true true` D. `false false`

E. `false true`

```
Object obj = new Object();
String str = "grace";
System.out.print( obj instanceof String );
System.out.print( " " );
System.out.print( str instanceof Object );
```

QUESTION 20

What is output by the code to the right?

A. `false` B. `true` C. `door`

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

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

```
String item = "door";
System.out.print( item.matches("d..r") );
```

QUESTION 21

What is output by the code to the right?

A. `[3, 7]` B. `[7, 3]` C. `[3]`

D. `[7, 0, 3]` E. `[0, 3, 7]`

```
ArrayList<Integer> nums
                  = new ArrayList<Integer>();
nums.add(7);
nums.add(0, 3);
System.out.print( nums );
```

QUESTION 22

Which of the following could replace **`<*1>`** in the code to the right as a syntactically legal identifier?

A. `value` B. `int`

C. `x+y` D. `num12`

E. More than one of these.

```
int <*1> = 42;
```

QUESTION 23

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

A. Duplicates may not be added to a `Set`.

B. `"B"` is a `char`, not a `String`.

C. Instances of interfaces cannot be created.

D. `Set`s cannot be iterated over using the enhanced `for` loop.

E. `Set`s cannot contain `String`s.

```
Set<String> smallSet = new Set<String>();
smallSet.add("A");
smallSet.add("B");
smallSet.add("A");
for( String str : smallSet )
  System.out.print( str );
```

QUESTION 24

What is output by the code to the right?

A. `X` B. `Y` C. `Z`

D. `ZY` E. `YX`

```
Queue<String> q = new LinkedList<String>();
q.add( "Z" );
q.add( "X" );
q.add( "Y" );
System.out.print( q.remove() );
```

QUESTION 25

What is output by the code to the right?

A. `8` B. `0` C. `6`

D. `7` E. `5`

```
int[] ary = {5, 7, 3};
int[] otherAry = ary;
otherAry[1]++;
otherAry = new int[5];
System.out.print( ary[1] );
```

QUESTION 26

How many `*`'s are output by the code to the right?

A. `27` B. `3` C. `10`

D. `30` E. `13`

```
for(int i = 0; i < 10; i++)
  for(int j = 0; j < 3; j++)
    System.out.print("*");
```

QUESTION 27

What replaces **<*1>** in the code to the right so that if the element at index `j` is less than the element at index `temp` according to their natural ordering, the statement `temp = j;` is executed?

A. `temp.compareTo( j ) <= 0`

B. `data[j] < data[temp]`

C. `data[j].compareTo( data[temp] ) == 0`

D. `j.compareTo( data[temp] ) > 0`

E. `data[j].compareTo( data[temp] ) < 0`

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

QUESTION 28

What replaces **<*2>** in the code to the right so that the elements originally at indices `i` and `j` in array `data` are swapped with each other?

A.
```
int t = i;
i = j;
j = t;
```

B.
```
Comparable t = data[i];
data[i] = data[j];
data[j] = t;
```

C.
```
data[i] = data[i] ^ data[j];
data[j] = data[i] ^ data[j];
data[i] = data[j] ^ data[i];
```

D.
```
data[i] = data[j];
data[j] = data[i];
```

E. More than one of these.

Assume **<*1>** and **<*2>** are filled in correctly.

QUESTION 29

What sorting algorithm is implemented by methods `sort` and `swap`?

A. Insertion sort B. Quick Sort

C. Selection Sort D. Shell Sort

E. Merge Sort

```
public static void sort(Comparable[] data){
  int temp;
  int len = data.length;
  for(int i = 0; i < len - 1; i++){
    temp = i;
    for(int j = i + 1; j < len; j++){
      if( <*1> )
        temp = j;
    }
    swap( data, i, temp);
  }
}

public static void swap(Comparable[] data,
                                int i, int j){
  <*2>
}
```

QUESTION 30

What replaces **<*1>** in the code to the right to indicate that the `TreeMap` named `encode` has `String`s for keys and `Integer`s for values?

A. `<Integer, String>`

B. `<String, int>`

C. `<int, String>`

D. `<String><int>`

E. `<String, Integer>`

```
TreeMap<*1> encode = new TreeMap<*1>();

encode.put("M", 212);
encode.put("A", 193);
encode.put("T", 227);


Iterator< Map.Entry<*1> > it;
it = encode.entrySet().iterator();
System.out.print( it.next().getValue() );
```

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

QUESTION 31

What is output by the code to the right?

A. `193` B. `M` C. `A`

D. `T` E. `227`

QUESTION 32

What is output by the code to the right when method `first` is called?

A. `1` B. `0`

C. `2` D. `5`

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

```
/* pre: data != null, elements of data are
sorted in ascending order.
*/
public static int find(int tgt, int[] data){
  int en = data.length - 1;
  return help(0, en, tgt, data);
}

private static int help(int st, int en,
int tgt, int[] data){
  int result = -1;
  int md, val;
  if( st <= en ){
      md = (st + en) / 2;
      val = data[ md ];
      if( val == tgt )
        result = md;
      else if( tgt < val )
        result =  help(st, md - 1, tgt, data);
      else
        result = help(md + 1, en, tgt, data);
  }
  return result;
}

public static void first(){
  int[] data = {0, 5, 19, 100};
  System.out.print( find(5, data) );
}
```

QUESTION 33

What searching algorithm is implemented by methods `find` and `help`?

A. linear search

B. interpolation search

C. random search

D. comb search

E. binary search

QUESTION 34

Given an array that contains N elements what is the expected running time of method `find`? Choose the most restrictive correct answer.

A. O(N) B. O(1) C. O(logN)

D. O(NlogN) E. O(sqrt(N))

QUESTION 35

What replaces **<*1>** in the code to the right so that method `isEmpty` returns `true` if the `ArrayList` `myCon` contains 0 elements?

A. `myCon.size() == 0 ? false : true`

B. `return size() > 0;`

C. `return super.size() == 0`

D. `return myCon.size() == 0`

E. `super.myCon.isEmpty();`

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

QUESTION 36

What is output by the code to the right when method `second` is called?

A. `CBA` B. `ABC` C. `CB`

D. `C` E. `CCC`

QUESTION 37

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

A. List B. Stack C. Queue

D. Heap E. Binary Search Tree

```
public class Structure<E>{

  private ArrayList<E> myCon;

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

  public void add(E obj){
    myCon.add(obj);
  }

  public E peek(){
    return myCon.get( myCon.size() - 1 );
  }

  public boolean isEmpty(){
    <*1>;
  }

  public E remove(){
    return myCon.remove(myCon.size() - 1);
  }
}

///////// client code /////////
public static void second(){
  Structure<String> s
                  = new Structure<String>();
  s.add( "A" );
  s.add( "B" );
  s.add( "C" );
  while( !s.isEmpty() )
    System.out.print( s.remove() );
}
```

QUESTION 38

Assume the method `sample(int[] data)` is $O(N^2)$ where N = `data.length`. When the method `sample` is passed an array with `length` = 100,000 it takes 2 seconds for method `sample` to complete. If method `sample` is then passed an array with `length` = 200,000 what is the expected time it will take method `sample` to complete?

A. 2 seconds B. 3 seconds C. 4 seconds D. 6 seconds E. 8 seconds

QUESTION 39

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

`2, 6, 1, 8, 0`

A. `2 1 0 6 8` B. `0 1 2 6 8` C. `0 1 8 6 2` D. `2 1 6 0 8` E. `0 8 1 6 2`

QUESTION 40

Which keyword is used in a method declaration to indicate the method may generate an exception, but will not try to handle it locally?

A. `try` B. `throws` C. `catch` D. `throw` E. `finally`

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 (to-1).
- String substring(int begin)
  Returns substring(from, length()).
- int indexOf(String str)
  Returns the index within this string of the first occurrence of the specified substring. Returns -1 if str is not found.
- int indexOf(String str, int fromIndex)
  Returns the index within this string of the first occurrence of the specified substring, 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 x.
- void add(int index, E e)
  Inserts x 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 Key
UIL Invitational A 2008

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

Notes:

22. Choices A and D are both syntactically legal identifiers.

31. The `TreeMap` stores keys in ascending order, thus the first entry in the map will be `["A", 193]` and `"A"` is the key for that entry.

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.