University Interscholastic League

# Computer Science Competition

Number 108 (Invitational B - 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. You may 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 $111_8$ and $777_8$?

A. $10000_8$ B. $8000_{10}$ C. $888_{10}$ D. $10100_8$ E. $1110_8$

**QUESTION 2**

What is output by the code to the right?

A. 10 B. 20 C. 14

D. 16 E. 0

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

**QUESTION 3**

What is output by the code to the right?

A. 21 B. 0 C. 20

D. 10 E. 40

```
int counter = 0;
for(int i = 0; i < 20; i++)
  counter++;
System.out.print( counter );
```

**QUESTION 4**

What is output by the code to the right?

A. 5 B. 0 C. 6

D. 1 E. -1

```
String subj = "mathematics";
System.out.print( subj.indexOf( 'm', 3 ) );
```

**QUESTION 5**

What is output by the code to the right?

A. 0.0 B. 8.0

C. 6.0 D. -4.0

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

```
double[] vals = {1.5, -1.0, 2.0};
vals[1] *= 4.0;
System.out.print( vals[1] );
```

**QUESTION 6**

What is output by the code to the right?

A. 9 B. 6 C. -9

D. 4 E. 1

```
int r = 3;
--r;
r *= r;
System.out.println( 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. `true false true false`

```
boolean p = true;
boolean q = false;
System.out.print( p && q );
System.out.print( " " );
System.out.print( p || q );
```

**QUESTION 8**

What is output by the code to the right?

A. `yno` B. `yn` C. `y`

D. `yo` E. `o`

```
int j = 10;
if( j < 10){
  if( 12 > j )
    System.out.print("y");
  else
    System.out.print("n");
}
else
  System.out.print("o");
```

**QUESTION 9**

What replaces **`<*1>`** in the code to the right so that the method `longSong` is accessible to code in all classes?

A. `private` B. `String` C. `void`

D. `public` E. `java.lang`

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

**QUESTION 10**

What replaces **`<*2>`** in the code to the right so the method `longSong` returns `true` only if the instance variable `lengthInSeconds` is greater than `180`?

A.
```
if( lengthInSeconds > 180 )
  return true;
else
  return false;
```

B.
```
if( lengthInSeconds != 180 )
  return true;
else
  return false;
```

C. `return lengthInSeconds > 180;`

D. `180.equals( lengthInSeconds );`

E. More than one of these.

```
public class Song{

  private String name;
  private int lengthInSeconds;

  public Song(String nm, int len){
    name = nm;
    lengthInSeconds = len;
  }

  <*1> boolean longSong(){
    <*2>
  }
}
```

**QUESTION 11**

What is output by the code to the right?

A. `true` B. `false` C. `0`

D. `16` E. `29`

```
int x = 13;
int y = 16;
System.out.print( x | y );
```

**QUESTION 12**

What is output by the code to the right?

A. `1` B. `2` C. `0`

D. `-2` E. `19`

```
System.out.print( Math.round(1.99) );
```

**QUESTION 13**

What is output by the code to the right?

A.
```
OneTwo
Three
```
B.
```
OneTwoThree
```
C.
```
One
Two
Three
```
D.
```
One
TwoThree
```
E.
```
Two
Three
```

```
System.out.println("One");
System.out.print("Two");
System.out.println("Three");
```

**QUESTION 14**

What is output by the code to the right?

A. `1.5` B. `1.50` C. `$2.00`

D. `$.50` E. `$1.50`

```
System.out.printf("$%2.2f", 1.5);
```

**QUESTION 15**

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

A. `3` B. `5` C. `4`

D. `7` E. `9`

```
public static int toy(int value){
  value++;
  value += 1;
  return value;
}
```

**QUESTION 16**

Which of the following replaces **`<*1>`** in the code to the right to convert `str` to an `int`?

A. `Integer.intValue()`

B. `num.toString(str)`

C. `Integer.parseInt(str)`

D. `Integer.compareTo(str)`

E. More than one of these.

```
String str = "-123";
int num = <*1>;
```

**QUESTION 17**

What is output by the code to the right?

A. `1455` B. `145` C. `5541`

D. `541` E. `5154`

```
int[] data = {5, 1, 5, 4};
Arrays.sort( data );
for( int i : data )
  System.out.print(i);
```

**QUESTION 18**

What is output by the code to the right?

A. `-13` B. `0` C. `13`

D. `-12` E. `-12.7`

```
double negValue = -12.7;
System.out.print( (int)negValue );
```

**QUESTION 19**

Which of the following method calls would return `true`?

I. `Character.isLetter( '8' )`

II. `Character.isDigit( '8' )`

III. `Character.isLetterOrDigit( '8' )`

A. I only B. II only C. III only D. I and II only E. II and III only

**QUESTION 20**

What is output by the code to the right?

A. `12` B. `EV` C. `OD`

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

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

```
int val = 12;
String stat = (val % 2 == 0) ? "EV" : "OD";
System.out.print( stat );
```

**QUESTION 21**

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

A. `0` B. `-1` C. `1`

D. `5` E. `3`

**QUESTION 22**

Which searching algorithm does method `find` implement?

A. Binary search

B. Stack search

C. Interpolation search

D. Gnome search

E. Sequential search

```
public static int find(int[] data,
                       int   tgt){
  int loc = -1;
  int i = 0;
  while( loc == -1 && i < data.length ){
    if( data[i] == tgt )
      loc = i;
    i++;
  }
  return loc;
}

public static void test(){
  int[] data = {3, 1, 5};
  System.out.print( find(data, 7) );
}
```

**QUESTION 23**

What replaces **`<*1>`** in the code to the right to generate an `Exception` if `data` is `null`?

A. `catch new IllegalArgumentException()`

B. `throw new IllegalArgumentException()`

C. `try new Error`

D. `try new IllegalArgumentException()`

E. `throws IllegalArgumentException()`

```
public static boolean evenLen(int[] data){
  if( data == null )
    <*1>;
  return data.length % 2 == 0;
}
```

**QUESTION 24**

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

A. `null:-1`

B. `null:0`

C. `:0`

D. `none:-1`

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

**QUESTION 25**

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

A. `Next:`

B. `Next:-1`

C. `Next:0`

D. `Next:null`

E. `Next:numSongs`

**QUESTION 26**

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

A. `false`

B. `true`

C. `null`

D. There is no output due to a syntax error in method `three.`

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

```
public class Album{
  private String title;
  private int numSongs;

  public Album(){
    this("none", -1);
  }

  public Album(String t){
    title = t;
  }

  public Album(String t, int num){
    title = t;
    numSongs = num;
  }

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

///////// client code /////////
public static void one(){
  Album a = new Album();
  System.out.print( a );
}

public static void two(){
  Album a = new Album("Next");
  System.out.print( a );
}

public static void three(){
  Album a1 = new Album();
  Album a2 = new Album();
  System.out.print( a1.equals(a2) );
}
```

**QUESTION 27**

What can replace the lines of code marked `line 1` and `line 2` in the code to the right without altering the output?

| | line 1 | line 2 |
|---|---|---|
| A. | `li.addFirst(1);` | `li.add(2);` |
| B. | `li.add(0,1);` | `li.addLast(2);` |
| C. | `li.addLast(1);` | `li.addLast(2);` |
| D. | `li.addLast(1);` | `li.addFirst(2);` |
| E. | `li.addFirst(1);` | `li.set(0, 2);` |

```
LinkedList<Integer> li;
li = new LinkedList<Integer>();
li.add(1); // line 1
li.add(0, 2); // line 2
System.out.print( li );
```

**QUESTION 28**

What replaces **<*1>** in the code to the right to obtain the character at position `i` in the `String s`?

A. `s[i]`

B. `charAt(s, i)`

C. `s.substring(i)`

D. `Character(s, i)`

E. `s.charAt(i)`

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

```
public static String myst(String s){
  String result = "";
  char ch;
  for(int i = 0; i < s.length(); i++){
    ch = <*1>;
    for(int j = 0; j <= i; j++)
      result = result + ch;
  }
  return result;
}
```

**QUESTION 29**

What is returned by the method call `myst("hot")`?

A. `hot`

B. `hoottt`

C. `ott`

D. `hhhooottt`

E. `hhhoot`

**QUESTION 30**

What will be the length of the `String` returned by method `myst` if the parameter `s` has a length of 20?

A. 20 B. 400 C. 210

D. 55 E. 20! (factorial of 20)

**QUESTION 31**

What is output by the code to the right?

A. `ads`

B. `sad`

C. `das`

D. `sda`

E. The output cannot be determined until run time.

```
TreeSet<Character> set;
set = new TreeSet<Character>();

set.add('s');
set.add('a');
set.add('d');

Iterator<Character> it = set.iterator();
while( it.hasNext() )
  System.out.print( it.next() );
```

**QUESTION 32**

Which sorting algorithm involves splitting the unsorted data into smaller and smaller parts and then recombining the parts into larger and larger sorted lists?

A. Quick sort B. Selection sort C. Insertion Sort D. Shell Sort E. Merge sort

**QUESTION 33**

What is output by the code to the right?

A. `2` B. `4` C. `24`

D. `213` E. `37`

```
Stack<Integer> s = new Stack<Integer>();
s.push(24);
s.push(213);
s.push(37);
System.out.print( s.peek() );
```

**QUESTION 34**

In the code to the right assume the `Collection col` contains N elements. What kind of `Collection` must `col` be so that each operation in method `demo` has an expected running time of O(1)?

A. `ArrayList`  B. `TreeSet`

C. `HashSet`  D. `LinkedList`

E. `ArrayMap`

```
// precondition: col does not contain 1000
public void demo(Collection<Integer> col){
  col.add( 1000 );
  boolean here = col.contains( 1000 );
  col.remove(1000);
}
```

**QUESTION 35**

What is output by the code to the right?

A. `9491`  B. `1949`  C. `1499`

D. `149`  E. `941`

```
PriorityQueue<Integer> pq;
pq = new PriorityQueue<Integer>();

pq.add(9);
pq.add(4);
pq.add(9);
pq.add(1);

while( !pq.isEmpty() )
  System.out.print( pq.remove() );
```

**QUESTION 36**

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

A. `22`  B. `1`  C. `4`

D. `15`  E. `3`

**QUESTION 37**

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

A. `63`  B. `0`  C. `5`

D. `127`  E. `1`

```
public class RecDemo{

  public int count;

  public int rec(int n){
    count++;
    if( n == 0 )
      return 1;
    else
      return 2 + rec(n - 1) + rec(n - 1);
  }
}

///////// client code /////////
public static void recOne(){
  RecDemo r = new RecDemo();
  System.out.print( r.rec(3) );
}

public static void recTwo(){
  RecDemo r = new RecDemo();
  r.count = 0;
  r.rec(5);
  System.out.print( r.count );
}
```

**QUESTION 38**

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

A. `0`

B. `null`

C. `-1`

D. There is no output due to a syntax error in method `structOne`.

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

**UESTION 39**

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

A. `317`

B. `3713`

C. `3173`

D. There is no output due to a syntax error in method `structTwo`.

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

**QUESTION 40**

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

A. A stack

B. A max heap

C. A queue

D. A binary search tree

E. A min heap

```
public class Structure<E>{

  private Stack<E> first;
  private Stack<E> second;

  public Structure(){
    first = new Stack<E>();
    second = new Stack<E>();
  }

  public void add(E item){
    first.push(item);
  }

  public E get(){
    if( second.isEmpty() )
      fill();
    return second.peek();
  }

  public E remove(){
    if( second.isEmpty() )
      fill();
    return second.pop();
  }

  public boolean isEmpty(){
    return first.isEmpty() &&
                          second.isEmpty();
  }

  private void fill(){
    while( !first.isEmpty() )
      second.push( first.pop() );
  }
}

///////// client code /////////
public static void structOne(){
  Structure<Integer> s;
  s = new Structure<Integer>();
  System.out.print( s.get() );
}

public static void structTwo(){
  Structure<Integer> s;
  s = new Structure<Integer>();

  s.add(3);
  s.add(1);
  s.add(7);
  s.add(3);

  while( !s.isEmpty() ){
    System.out.print(  s.remove() );
  }
}
```

**No 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()`

**`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 Key
# UIL Invitational B 2008

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

**Notes:**

10. Answer E. Choices A and C are both correct.

26. A. The `Album` class inherits the `equals` method from the `Object` class. This method returns true of the calling object is referring to the same object as the explicit parameter. It does not check any instance variables. In other words: `return this == other;`

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.