University Interscholastic League

# Computer Science Competition

Number 113 (Invitational A - 2009)

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 is the sum of $10101_2$ and $111_2$?

A. $11100_2$ B. $1110_2$ C. $11111_2$ D. $11000_2$ E. $10001_2$

**QUESTION 2**

What is output by the code to the right?

A. 5 B. 7 C. 8

D. -3 E. 6

```
int x = 5;
int y = x + 2;
x++;
System.out.println( y );
```

**QUESTION 3**

What is output by the code to the right?

A. 16 B. 0 C. 32

D. 10 E. 15

```
int count = 0;
for(int i = 0; i < 15; i++){
  count++;
}
System.out.print( count );
```

**QUESTION 4**

What is output by the code to the right?

A. matlab B. ma C. mat

D. atlab E. tlab

```
String lang = "matlab";
String res = lang.substring(2);
System.out.println( res );
```

**QUESTION 5**

What is output by the code to the right?

A. 5 B. 4 C. 2

D. 3 E. 6

```
int[] nums = {5, 12, -7, 4, 5, 2};
System.out.print( nums[4] );
```

**QUESTION 6**

What is output by the code to the right?

A. 0 B. 0.25 C. 2.5

D. 4.5 E. 5.0

```
double a = 2.5;
a += 5 / 2;
System.out.print( a );
```

**QUESTION 7**

How many combinations of values for the `boolean` variables `a`, `b`, and `c` will result in `d` being set to `true`?

A. 1 B. 4 C. 7

D. 0 E. 3

```
boolean a, b, c;
//code to initialize a, b, and c

boolean d = ( a || b || c );
```

**QUESTION 8**

What is output by the code to the right?

A. `2` B. `12` C. `123`

D. `13` E. `3`

```
int u = 5;
double v = 5.5;
if( v > u )
  System.out.print( 1 );
if( (int)v > u )
  System.out.print( 2 );
else
  System.out.print( 3 );
```

**QUESTION 9**

What replaces **<*1>** in the code to the right so that the `Pizza` class inherits from the `Food` class?

A. `implements`

B. `inherits`

C. `final`

D. `static`

E. `extends`

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

**QUESTION 10**

What replaces **<*2>** in the code to the right to concatenate to result the value stored in the variable `cost`?

I. `super.cost`
II. `super.getCost()`
III. `getCost()`

A. I only

B. II only

C. III only

D. Both I and II

E. Both II and III

```
public class Food{
  private int cost;

  public Food(int c){
    cost = c;
  }

  public int getCost(){
    return cost;
  }
}

public class Pizza <*1> Food{
  private int size;

   public Pizza(int cost, int sz){
    super(cost);
    size = sz;
  }

  public String toString(){
    String result = "sz: " + size;
    result += ", cost: " + <*2>;
    return result;
  }
}
```

**QUESTION 11**

What is output by the code to the right?

A. `47` B. `32` C. `true`

D. `15` E. `false`

```
int m = 15;
int n = 32;
System.out.print( m | n );
```

**QUESTION 12**

What is output by the code to the right?

A. `9` B. `9.0`

C. `4` D. `4.0`

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

```
int num = 3;
System.out.print( Math.pow(num, num - 1) );
```

**QUESTION 13**

What is output by the code to the right?

A.
```
One
Two
Three
```

B.
```
One
TwoThree
```

C.
```
OneTwo
Three
```

D.
```
123
```

E.
```
OneTwoThree
```

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

**QUESTION 14**

What is output by the code to the right?

A. `33.10` B. `33.1000` C. `33.100`

D. `4.2f` E. `33.10000`

```
System.out.printf("%4.2f", 33.1);
```

**QUESTION 15**

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

A. `3` B. `16` C. `0`

D. `10` E. `34`

```
public int basic(int x){
  x = x * x;
  x++;
  return x;
}
```

**QUESTION 16**

The code to the right does not compile. Which line contains the syntax error?

A. `Line A` B. `Line B`

C. `Line C` D. `Line D`

E. `Line E`

```
Object[] things = new Object[10];
things[0] = "cat"; // line A
things[1] = new Integer(12); // line B
things[12] = new Character('A'); // line C
char c2 = things[0].charAt(1); // line D
System.out.print( things[1] );  // line E
```

**QUESTION 17**

What are the possible outputs for the code to the right?

I. `0`
II. `1`
III. `3`

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

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

```
int x, y, z;
// code to initialize x, y, and z
String result = "";
if ( x > 10 )
  result += "a";
else if( y > 10 )
  result += "a";
else if( z > 10 )
  result += "a";
else
  result += "a";
System.out.println( result.length() );
```

**QUESTION 18**

What is output by the code to the right?

A. `DC3DC3NY` B. `DC` C. `DCDCNY`

D. `3DC3DCNY` E. `DCDCDCDCDCDCNY`

```
String temp = "DC";
temp += 3;
temp += temp + "NY";
System.out.print( temp );
```

**QUESTION 19**

What is output by the code to the right?

A. 0 B. 3 C. 9

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

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

```
String[] subjs = {"Chem", "Bio", "CS"};
int total = 0;
for( String st : subjs )
  total += st.length();
System.out.print( total );
```

**QUESTION 20**

What replaces **<*1>** in the code to the right so that the value stored in `LIMIT` cannot be changed?

A. `local` B. `const` C. `static`

D. `final` E. `this`

```
public int check(int x){
  <*1> double LIMIT = Math.sqrt(x);

  // rest of the method not shown
}
```

**QUESTION 21**

What is output by the code to the right?

A. 12 B. 11 C. 14

D. 8 E. 10

```
int v = 5;
int w = --v * 2;
System.out.print(w);
```

**QUESTION 22**

What is output by the client code to the right?

A. 6 B. 2 C. 10

D. 5 E. 4

**QUESTION 23**

Which searching algorithm does method `indexOf` use?

A. `Binary` B. `Quick` C. `Merge`

D. `Bubble` E. `Sequential`

```
public int indexOf(int start, int[] data,
                                    int tgt){
  int result = -1;
  for(int i = start; i < data.length; i++){
    if( tgt == data[i] ){
      result = i;
      break;
    }
  }
  return result;
}

//client code
int[] scores = {0, -5, 10, 240, 10, 10};
System.out.print( indexOf(3, scores, 10) );
```

**QUESTION 24**

What is output by the code to the right?

A. `[3, 3]` B. `[2, 0, 2]`

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

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

```
ArrayList<Integer> laps;
laps = new ArrayList<Integer>();
laps.add(2);
laps.add(0, 1);
laps.add(0, 3);
laps.set(2, laps.get(1) );
System.out.print( laps );
```

**QUESTION 25**

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

A. 10 B. 9 C. 12

D. 5 E. 14

```
public int many(int n){
  if(n == 1)
    return 5;
  else
    return n + many(n - 1);
}
```

**QUESTION 26**

What is output by the code to the right when given this input?

```
5.2.1.2\2\2.2
```

A. `10` B. `16` C. `14`

D. `5` E. `8`

```
Scanner sc = new Scanner(System.in);
sc.useDelimiter("\\.");

int sum = 0;
while( sc.hasNextInt() )
  sum += sc.nextInt();
System.out.print( sum );
```

**QUESTION 27**

What replaces **<*1>** in the code to the right so method `numVowels` generates an exception and ends if the precondition is not met?

A. `return` B. `catch` C. `end`

D. `throw` E. `assert`

```
// pre: s != null, s.length() > 0
public boolean same(String s){
  if( s == null || s.length() <= 0 )
    <*1> new IllegalArgumentException();

  int last = s.length() - 1;
  return s.charAt(0) == s.charAt(last);
}
```

**QUESTION 28**

Which sorting algorithm does method `sort` implement?

A. Quicksort B. Insertion sort

C. Merge sort D. Radix sort

E. Selection sort

**QUESTION 29**

What is the Big O of method `sort` given an `ArrayList` of `Integer`s already sorted in descending order? Pick the most restrictive correct answer.

A. $O(N^2)$ B. $O(N)$

C. $O(1)$ D. $O(N^3)$

E. $O(N\log N)$

```
// post: sort elements into desending order
public void sort(ArrayList<Integer> data){
  int max, temp;
  int lim = data.size() - 1;
  for(int i = 0; i < lim; i++){
    max = i;
    for(int j = i + 1; j < data.size(); j++)
      if( data.get(j) > data.get(max) )
        max = j;
    if( i != max ){
      temp = data.remove(i);
      data.add(i, data.remove(max - 1) );
      data.add(max, temp);
    }
  }
}
```

**QUESTION 30**

Which of the following can replace **<*1>** in the code to the right without causing a syntax error?

A. `new HashSet<Integer>()`

B. `new List<Integer>()`

C. `new LinkedList<Integer>()`

D. `new int[10]`

E. More than one of these are correct.

```
List<Integer> times;
times = <*1>;
```

**QUESTION 31**

What is output by the code to the right?

A. `12` B. `5` C. `7`

D. `8` E. `null`

```
PriorityQueue<Integer> pq;
pq = new PriorityQueue<Integer>();
int[] toAdd = {12, 5, 7, 5, 8};

for(int i : toAdd)
  pq.add(i);
pq.remove();
System.out.println( pq.peek() );
```

**QUESTION 32**

What is output by the code to the right?

A. `obj` B. `null`

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

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

E. The output can not be determined until runtime.

```
Object obj = new Object();
System.out.println( obj.toString() );
```

**QUESTION 33**

If N equals `rds.length` what is the Big O of method `process`? Pick the most restrictive correct answer.

A. O(NlogN) B. O(N!) C. $O(N^2)$

D. $O(N^2 \log N)$ E. $O(N^3)$

```
public int process(int[] rds){
  int total = 0;
  int lim = rds.length;
  for(int i = 0; i < lim; i++){
    for(int j = 1; j < lim; j *= 2)
      total += rds[i] * rds[j];
    for(int j = i; j < lim; j++)
      total += rds[i] + rds[j];
  }
  return total;
}
```

**QUESTION 34**

The height of a tree is the number of links from the root of the tree to the deepest leaf in the tree. The following values are inserted one at a time in the order shown into a binary search tree using the traditional insertion algorithm. What is the height of the resulting tree?

`12, 52, 100, 13, 50, -10`

A. `0` B. `1` C. `3` D. `4` E. `2`

**QUESTION 35**

What is output by the code to the right?

A. `[5]`

B. `[5, 11]`

C. `[5, 7, 13]`

D. `[13, 5, 7]`

E. There is no way to determine the output until runtime.

```
TreeSet<Integer> t1 = new TreeSet<Integer>();
TreeSet<Integer> t2 = new TreeSet<Integer>();
t1.add(5);
t1.add(11);
t2.add(13);
t2.add(5);
t2.add(7);
t1.retainAll(t2);
System.out.println(t1);
```

**QUESTION 36**

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

A. `14` B. `11` C. `17`

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

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

**QUESTION 37**

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

A. `22` B. `19` C.

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

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

```
public int sum(int[][] mat, int pos){
  int result = 0;
  for(int i = 0; i < mat.length; i++){
    result += mat[i][pos];
    result += mat[pos][i];
  }
  return result;
}

public void testA(){
  int[][] mA = {{5,1,3},{1,5,3},{1,2,2}};
  System.out.print( sum(mA, 1) );
}

public void testB(){
  int[][] mB = {{3,1,4},{1,6,2},{2,3,5},
                                    {2,5,1}};
  System.out.print( sum(mB, 2) );
}
```

**QUESTION 38**

What replaces **<*1>** in the code to the right to make that block of code the default constructor for the `Structure` class?

A. `Structure()` B. `new Structure()`

C. `()` D. `Structure`

E. `void Structure()`

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

**QUESTION 39**

What is output by the following client code?

```
Structure s = new Structure();
s.add("A");
System.out.println( s.get(0) );
```

A. `A` B. `null`

C. `start` D. `temp`

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

**QUESTION 40**

What type of data structure do the `Node` and `Structure` classes implement?

A. A linked list B. A binary tree

C. A min heap D. A hash table

E. A max heap

```
public class Node{
  public Object data;
  public Node next;
}

public class Structure{
  private Node start;

  public <*1>{
    start = new Node();
  }

  public void add(Object obj){
    Node temp = start;
    while( temp.next != null )
      temp = temp.next;
    temp.next = new Node();
    temp.next.data = obj;
  }

  public Object get(int pos){
    Node temp = start.next;
    for(int i = 0; i < pos; i++)
      temp = temp.next;
    return temp.data;
  }

  public void remove(int pos){
    Node temp = start;
    for(int i = 0; i < pos; i++)
      temp = temp.next;
    temp.next = temp.next.next;
  }
}
```

# 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 A 2009

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

**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.

6. The expression `5/2` results in integer division because both operands are `int`s. The result is `2` which is then converted to `2.0` and that is added to `a` resulting in a holding the value `4.5`.

10. A descendant class does not have access to the private instance variables in ancestor classes. Choice I is not correct.

16. The declared type of things[0] is `Object`. The `Object` class does not have a `charAt` method so line D results in a syntax error. Line C will cause a runtime error, but it is caught at compile time.

29. The outer loop executes N times. Inside the loop there are 5 elements that are each O(N). The inner loop, the two calls to remove, and the two calls to add. So the work of the body of the inner loop is 5N. Even though the approach used to swap values is slow in it does not change the overall Big O of the algorithm.

32. The `toString` method in the `Object` class prints out the class name and the object's `hashcode`. The result `Object's hashcode` method "is typically implemented by converting the internal address of the object into an integer" which means the output can vary from one run of the program to the next. Thus the output cannot be determined at compile time.