University Interscholastic League

Computer Science Competition

Number 110 (District 2 - 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 is the sum of 757_8 and 363_8?

A. 1342_8 B. 1120_8 C. 1777_8 D. 1111_8 E. 1130_8

QUESTION 2

What is output by the code to the right?

A. `1.0` B. `1.5` C. `1`

D. `0` E. `0.0`

```
double a = 3.0;
double b = 2.0;
System.out.print( a / b );
```

QUESTION 3

What is output by the code to the right?

A. `13` B. `0` C. `12`

D. `14` E. `28`

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

QUESTION 4

What is output by the code to the right?

A. `en_co` B. `hen_` C. `hen`

D. `enco` E. `en_c`

```
String sci = "stephen_cook";
String some = sci.substring(5, 9);
System.out.print( some );
```

QUESTION 5

What is output by the code to the right?

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

D. `3` E. `13`

```
int[] values = {3, 5, 2, 12, 7};
values[2]++;
System.out.print( values[2] );
```

QUESTION 6

What is output by the code to the right?

A. `10` B. `16` C. `18`

D. `8` E. `9`

```
int x = 5;
int y = 2;
y = x * (y + 2) - y;
System.out.print( y );
```

QUESTION 7

What is output by the code to the right?

A. `false false`

B. `false true`

C. `true false`

D. `true true`

E. `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. `5.0` B. `3.5` C. `6.0`

D. `4.0` E. `2.5`

```
double c = 2.5;
if( c > 1.5 ){
  if( c > 3 )
    c = c * 2;
  else
    c = c + 1;
}
System.out.print( c );
```

QUESTION 9

What replaces **`<*1>`** in the code to the right so that `DEFAULT_SIDES` is a class constant that is accessible in all other classes?

A. `public final`

B. `static final`

C. `public static`

D. `public static final`

E. `public static final void`

```
public class NumberDie{

  <*1> int DEFAULT_SIDES = 6;

  private int sides;

  public NumberDie(){
    sides = DEFAULT_SIDES;
  }

  public NumberDie(int s){
    sides = s;
  }

  public int getSides(){
    return sides;
  }
}

//////////////////////////////////////////
// client code
NumberDie d = new NumberDie();
System.out.print( d.getSides() );
```

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

QUESTION 10

What is output by the client code to the right?

A. `0` B. `d` C. `s`

D. `5` E. `6`

QUESTION 11

What is output by the code to the right?

A. `15` B. `7` C. `22`

D. `0` E. `1`

```
int f = 7;
int g = 15;
System.out.print( g & f );
```

QUESTION 12

What is output by the code to the right?

A. `0` B. `2` C. `7`

D. `14` E. `21`

```
int top = Math.max(7, 14);
System.out.print( top );
```

QUESTION 13

What is output by the code to the right?

A. `\Easy` B. `"easy"` C. `Easy`

D. `"Easy` E. `\\Easy`

```
System.out.print("\\Easy");
```

QUESTION 14

What is output by the code to the right?

A. `.3679` B. `3679` C. `1.3600`

D. `1.37` E. `+1.3`

```
System.out.printf("%4.2f", 1.3679);
```

QUESTION 15

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

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

D. `16` E. `6`

```
public static int sample(int y){
  y--;
  return y * y;
}
```

QUESTION 16

What is output by the code to the right?

A. `*=` B. `can`

C. `putt` D. `-`

E. There is no output.

```
String trash = "ham+can--putt*=cut";
String[] sp = trash.split("\\W+");
System.out.print( sp[2] );
```

QUESTION 17

What is output by the code to the right?

A. `0` B. `1` C. `3`

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

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

```
Object obj = new Object();
String st = obj;
System.out.print( st.length() );
```

QUESTION 18

What is output by the code to the right?

A. `9` B. `0`

C. `3` D. `6`

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

```
int x = 3;
int y = 3;
char[] letters = new char[ x * y ];
System.out.print( letters.length );
```

QUESTION 19

What is output by the code to the right?

A. `3`

B. `2`

C. `1`

D. `0`

E. `-1`

```
Double d1 = new Double( -1.23 );
Double d2 = new Double( 1.5 );
int comp = d2.compareTo( d1 );

if( comp < 0 )
  System.out.print( 1 );
else if ( comp == 0 )
  System.out.print( 2 );
else
  System.out.print( 3 );
```

QUESTION 20

What replaces **`<*1>`** in the code to the right to set the new `Pizza` object's `cost` instance variable equal to `10`?

A. `super(10)`

B. `Pizza(10)`

C. `this.Pizza(10)`

D. `this(10)`

E. `c = 10`

```
public class Pizza{
  private int cost;

  public Pizza(){
    <*1>;
  }

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

  public int getCost(){
    return cost;
  }
}

public class SquarePizza extends Pizza{

  private int sideLength;

  public SquarePizza(int sz){
    sideLength = sz;
  }

  public int pizzaPerDollar(){
    int area = sideLength * sideLength;
    return area / getCost();
  }
}

//////////////////////////////////////////
// client code
SquarePizza pz = new SquarePizza(20);
System.out.print( pz.pizzaPerDollar() );
```

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

QUESTION 21

What is output by the client code to the right?

A. `0`

B. `40`

C. `10`

D. `20`

E. There is no output due to a syntax error in the `SquarePizza` class.

QUESTION 22

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

`1.2 0.9 .3 -.4 ZZ`

A. `2` B. `2.1` C. `4`

D. `3` E. `2.0`

```
Scanner sc = new Scanner( System.in );
int count = 0;
while( sc.hasNextDouble() ){
  count++;
  sc.nextDouble();
}

System.out.print( count );
```

QUESTION 23

What is output by the code to the right?

A. `false`

B. `false false`

C. `false true`

D. `true false`

E. `true true`

```
String thing = "cat";
System.out.print( thing.matches( "ca" ) );
System.out.print( " " );
System.out.print( thing.matches( "c.*" ) );
```

Question 24

What is output by the code to the right?

A. `[1, 0, 3]` B. `[0, 1, 3]`

C. `[3, 1, 0]` D. `[0, 3, 1]`

E. `[1, 3, 0]`

```
ArrayList<Integer> ar;
ar = new ArrayList<Integer>();

ar.add(3);
ar.add(0, 1);
ar.add(1, 0);

System.out.print( ar.toString() );
```

Question 25

Which of the following are Java keywords?

I. `method`
II. `throws`
III. `foreach`

A. I only B. II only C. III only D. II and III E. none of these

Question 26

What is output by the code to the right?

A. `7`

B. `11`

C. `8`

D. `6`

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

```
String vals = "ab12CC3";
int pos = 0;
int cnt = 0;
boolean ts;

do{
  ts = Character.isLetter(vals.charAt(pos));
  if( ts ){
    cnt++;
    pos += 2;
  }
  else{
    cnt += 2;
    pos++;
  }
}while( pos < vals.length() );

System.out.print( cnt );
```

Question 27

What is returned by the method call `again(1)`?

A. `4` B. `-1`

C. `2` D. `0`

E. `1`

```
public static int again(int x){
  int result = 0;
  if( x <= 0 )
    result = 2;
  else
    result = 2 + again( x - 1 );
  return result;
}
```

Question 28

What is the running time of method `process`? Assume N equals `data.length`. Choose the most restrictive correct answer.

A. O(N^2) B. O(NlogN) C. O(N^3)

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

```
public static void process(int[] data,
                           int min,
                           int max){
  int result = 0;
  for( int val : data )
    if( val >= min && val <= max )
      result++;
    else
      result--;

  for(int i = 0; i < data.length; i++)
    data[i] += result;
}
```

QUESTION 29

Which searching algorithm does method `search` implement?

A. sequential search

B. insertion search

C. binary search

D. merge search

E. fibonacci search

QUESTION 30

What must the pre-condition for method `search` be so that it always fulfills its post-condition?

A. The elements of `data` must be in ascending order.

B. The elements of `data` must be in descending order.

C. The elements of `data` must be distinct. There can't be any duplicated values.

D. The length of `data` must be even.

E. More than one of these.

```
/* pre: see question 30
   post: return an index in data
         that contains tgt.
         If tgt is not present return -1.
*/
public static int search(int tgt,
                         int[] data){
  int high = data.length - 1;
  int low = 0;
  int mid = (low + high) / 2;

  while( data[mid] != tgt && low <= high){
    if( data[mid] > tgt )
      low = mid + 1;
    else
      high = mid - 1;
    mid = (low + high) / 2;
  }

  if( data[mid] == tgt )
    return mid;
  else
    return -1;
}
```

QUESTION 31

What is returned by the method call `one(2)`?

A. `0` B. `2` C. `4`

D. `3` E. `-1`

QUESTION 32

What is returned by the method call `three(1)`?

A. `3` B. `2` C. `5`

D. `6` E. `4`

```
public static int one(int a){
  return a * 2;
}

public static int two(int b){
  b = one(b + 1);
  return b;
}

public static int three(int c){
   c = two(c);
   c += 2;
   return c;
}
```

QUESTION 33

What are the elements in the `Set` named `beta` after the code to the right executes?

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

B. `[-2, -1, 0, 0, 1, 1, 2, 2, 3]`

C. `[0, 1, 2]`

D. `[3]`

E. `[-2, -1, 0, 1, 2, 3]`

```
Set<Integer> alpha;
alpha = new TreeSet<Integer>();
Set<Integer> beta = new TreeSet<Integer>();

for(int i = 2; i > -3; i--)
  alpha.add(i);

for(int i = 0; i < 4; i++)
  beta.add( i );

beta.addAll(alpha);
```

QUESTION 34

What replaces **<*1>** in the code to the right to set the variable `pos` equal to the number of elements in the `ArrayList` named `con`?

A. `con.size()`

B. `con.length()`

C. `con.size`

D. `super.size()`

E. `this.size()`

```
public class Structure
                 <E extends Comparable<E>>{

  private ArrayList<E> con;

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

  public boolean isEmpty(){
    return con.size() == 0;
  }

  public E get(){
    return con.get(0);
  }

  public E remove(){
    return con.remove(0);
  }

  public void add(E obj){
    int pos = <*1>;
    boolean done = false;
    E temp;
    while( pos > 0 && !done ){
      temp = con.get( pos - 1 );
      done = obj.compareTo( temp ) >= 0;
      if( !done )
        pos--;
    }
    con.add(pos, obj);
  }

  public String toString(){
    return con.toString();
  }
}

//////////////////////////////////////////
// client code
Structure<String> s;
s = new Structure<String>();
s.add("Z");
s.add("A");
s.add("Z");
s.add("B");
System.out.println(s);
```

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

QUESTION 35

What is output by the client code to the right?

A. `[Z, Z, B, A]`

B. `[A, B, Z]`

C. `[A, B, Z, Z]`

D. `[Z, B, A]`

E. `[Z, A, Z, B]`

QUESTION 36

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

A. A stack

B. A list

C. A priority queue

D. A set

E. A regular queue

QUESTION 37

In the code to the right, what kind of `Collection` must `col` be so that no `int` appears more than once?

A. `ArrayList` B. `Stack` C. `Queue`

D. `LinkedList` E. `HashSet`

```
public void show(Collection<Integer> col){
  for(int i : col)
    System.out.print( i );
}
```

QUESTION 38

What is output by the code to the right?

A. `true`

B. `false`

C. The output cannot be determined until runtime.

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

E. There is no output due to `ClassCastException`.

```
LinkedList<Integer> first;
first = new LinkedList<Integer>();
first.add( 12 );
first.add( 16 );

ArrayList<Integer> second;
second = new ArrayList<Integer>();
second.add( 12 );
second.add( 16 );

System.out.print( first.equals(second) );
```

QUESTION 39

Which of the following is most likely to occur when the program `RunLong` is compiled and run on a real computer?

A. The program will not compile due to a syntax error.

B. The program will run forever with no output.

C. The program will run forever printing out always increasing values.

D. The program will run forever printing out always decreasing values.

E. The program will eventually crash when it runs out of memory.

```
public class RunLong{

  public static void main(String[] args){
    go(1);
  }

  public static void go(int x){
    go( x + 1 );
    System.out.println( x );
  }
}
```

QUESTION 40

Class `C` to the right will not compile due to a syntax error. Which of the following best explains the syntax error?

A. Class `C` cannot extend two classes.

B. Class `A` and class `B` contain an instance variable with the same name.

C. Classes with `public` instance variables cannot be extended.

D. Class `C` does not override the `toString` method.

E. Class `C` must include a constructor.

```
public class A{
  public int x;
}

public class B{
  public int x;
}

public class C extends A, B{
  public int y;
}
```

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 District 2 2008

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

16. The Pattern `\W+` uses one or more non-word characters as delimiters. All characters besides a-zA-z_0-9 are used as delimiters.

37. A `HashSet` is the only data structure listed that does not have the possibility of duplicate elements in the collection.

38. The Java `ArrayList` and `LinkedList` classes both use the `equals` method from the `AbtsractList` class. An `ArrayList` and a `LinkedList` are equal if they are the same size and contain equivalent elements in the same order.