University Interscholastic League

Computer Science Competition

Number 115 (District 1 - 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 512_8 and 177_8?

A. 601_8 B. 701_8 C. 711_8 D. 611_8 E. 612_8

QUESTION 2

What is output by the code to the right?

A. 10.0 B. 7.0 C. 2.5

D. 9.0 E. 8.0

```
double a = 2.5;
double b = 2.0;
a *= b + 2;
System.out.println( a );
```

QUESTION 3

What is output by the code to the right?

A. 12 B. 20 C. 2

D. 22 E. 11

```
int sum = 0;
for(int i = 1; i < 12; i++){
  sum += 2;
}
System.out.print( sum );
```

QUESTION 4

What is output by the code to the right?

A. A B. B C. 1

D. 0 E. -1

```
String s1 = "A";
String s2 = "B";
System.out.print( s1.compareTo( s2 ) );
```

QUESTION 5

What is output by the code to the right?

A. 2 B. 0 C. 7

D. 1 E. -1

```
int[] scs = {3, 1, 0, 2, 3, 0, 1};
System.out.print( scs[ scs[0] ] );
```

QUESTION 6

What is output by the code to the right?

A. 4 B. 7.5 C. 3

D. 2 E. 7

```
int r = 3;
int s = 2;
int t = r * s + r / s;
System.out.print( t );
```

QUESTION 7

What is output by the code to the right?

A. false false

B. false true

C. true false

D. true true

E. true false true false

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

QUESTION 8

What is output by the code to the right?

A. 12 B. 2 C. 1

D. 21 E. 212

```
double m = 1.5;
double n = 2.5;
if( m > n )
  n *= 2;
else
  m *= 2;
if( m > 2 )
  System.out.print( 1 );
else
  System.out.print( 2 );
```

QUESTION 9

Consider the `Person` class and client code to the right . What is output by the statement marked `line 1`?

A. 0_0

B. null_null

C. 150_70

D. 70_150

E. p1

QUESTION 10

Consider the `Person` class and client code to the right. What is output by the statement marked `line 2`?

A. 0_0

B. p2

C. null_null

D. 54_150

E. 54_0

```
public class Person{
  private int height;
  private int weight;

  public Person(){
    this(70, 150);
  }

  public Person(int h){
    height = h;
  }

  public Person(int h, int w){
    height = h;
    weight = w;
  }

  public String toString(){
    return height + "_" + weight;
  }
}

////////////////////////////////////////
// client code
Person p1 = new Person();
System.out.println( p1 ); // line 1

Person p2 = new Person(54);
System.out.println( p2 ); // line 2
```

QUESTION 11

What is output by the code to the right?

A. 14 B. 58 C. -58

D. 3364 E. 232

```
int m = 58;
int n = m >> 2;
System.out.print( n );
```

QUESTION 12

What is output by the code to the right?

A. 5 B. 2 C. 0

D. 20 E. 10

```
int x = 10;
System.out.print( Math.max(x, (x / 2)) );
```

QUESTION 13

What is output by the code to the right?

A. `AlanKay` B. `AlannKay`

C. `AlanKAY` D. `Alan`
`Kay`

E. `Alan Kay`

```
String name = "Alan\nKay";
System.out.print( name );
```

QUESTION 14

What is output by the code to the right?

A. `275.000` B. `275` C. `+300`

D. `+275` E. `+000275`

```
System.out.printf("%+3d", 275);
```

QUESTION 15

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

A. `-5` B. `3` C. `-3`

D. `5` E. `-2`

```
public int process(int z){
  final int LOCAL = z * 2;
  z++;
  z = z + LOCAL;
  return z;
}
```

QUESTION 16

What is output by the code to the right?

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

D. `7` E. `4`

```
String stuff = "two three five seven";
String[] words = stuff.split("\\s+");
System.out.print( words.length );
```

QUESTION 17

What is output by the code to the right?

A. `0123` B. `1234` C. `123`

D. `0000` E. `1123`

```
int[] fibs = {1, 1, 2, 3};
for(int i : fibs)
  System.out.print( i );
```

QUESTION 18

What replaces **<*1>** in the code to the right so that the code segment compiles without error?

A. `(String)` B. `(Object)`

C. `(length)` D. `(toString)`

E. More than one of these is correct.

```
Object obj = "Sam";
int len = (<*1> obj).length();
```

QUESTION 19

What is returned by the method call `recurs(7)`?

A. `10` B. `2` C. `4`

D. `7` E. `20`

```
public int recurs(int n){
  int result = 0;
  if( n <= 3 )
    result = 2;
  else
    result = recurs(n - 2) + (n - 2);
  return result;
}
```

QUESTION 20

What is output by the code to the right?

A. `bfb`

B. `bbfbb`

C. `b`

D. `bbbfbbbb`

E. `bfbb`

```
for(int i = 8; i < 13; i++){
  if( i % 3 != 0 && i % 5 != 0 )
    continue;
  if( i % 5 == 0 )
    System.out.print('f');
  System.out.print('b');
}
```

QUESTION 21

What is output by the client code to the right?

A. `-3`

B. `-8`

C. `-6`

D. `0`

E. `-7`

```
public int off(int month){
  int result = -4;
  switch( month ){
    case 1: result = -3; break;
    case 3: case 5: case 8: case 10:
      result = -1; break;
    default: result = 0;
  }
  return result;
}

// client code
System.out.print( off(1) + off(7) );
```

QUESTION 22

What is output by the code to the right?

A. `0` B. `10` C. `null`

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

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

```
List<String> titles = new List<String>();
System.out.print( titles.size() );
```

QUESTION 23

What is output by the code to the right?

A. `1` B. `2` C. `12`

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

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

```
int[] ps = {2, 3, 5, 7, 11};
if( ps[3] < ps.length && ps[ps[3]] > 0 )
  System.out.print( 2 );
else
  System.out.print( 1 );
```

QUESTION 24

Which of the following best describes the purpose of an `Iterator` object?

A. Provide a way to insert elements into a data structure.

B. Provide access to the `private` instance variables of a data structure and a way to change their capacity.

C. Provide a standard way to access the elements of a data structure one element at a time.

D. Provide a way for data structures to hold any type of object.

E. Provide a way to sort all the elements of a data structure.

QUESTION 25

What replaces **<*1>** in the code to the right so that the body of the `while` loop is skipped if `char c` has been found in `String s`?

A. `result`

B. `!result`

C. `result == -1`

D. `result != -1`

E. `continue`

```
public int findChar(String s,
                    char c,
                    int start){
  int result = -1;
  int index = start;
  while( <*1> && index < s.length() ){
    if( s.charAt(index) == c )
      result = index;
    index++;
  }
  return result;
}
```

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

QUESTION 26

Which searching algorithm does method `findChar` use?

A. hash B. binary C. tree

D. heap E. sequential

QUESTION 27

Which of the following is a Java keyword?

A. `do` B. `foreach` C. `trys` D. `extra` E. `args`

QUESTION 28

What is output by the code to the right?

A. `true` B. `false` C. `2`

D. `1` E. `0`

```
int x = 3;
int y = 5;
if( (x > y) && (x == y) || (x * 2 > y) )
  System.out.print(1);
else
  System.out.print(2);
```

QUESTION 29

Consider method divide to the right. When the code is executing, if the lines marked `Point A` and `Point B` are reached, is the Boolean expression `n % 3 == 0` never, sometimes, or always true at those points?

	`Point A`	`Point B`
A.	Always	Always
B.	Always	Never
C.	Sometimes	Sometimes
D.	Sometimes	Never
E.	Always	Sometimes

```
public void divide(int n){
  if( n > 0 ){
    while( n % 3 == 0 ){
      // Point A
      n = n / 3;
      // Point B
    }
  }
  System.out.print( n );
}
```

QUESTION 30

In the code to the right how many times is the Boolean expression `i < vals.length` evaluated?

A. `vals.length`^{2} B. `vals.length - 1`

C. `vals.length` D. `vals.length + 1`

E. `vals.length / 2`

```
// pre: vals.length > 0
public int look(int[] vals, int find){
  int count = 0;
  for(int i = 0; i < vals.length; i++){
    count++;
    if( vals[i] == find )
      count--;
  }
  return count;
}
```

QUESTION 31

Assume `vals.length` is even. If exactly half of the elements in `vals.length` are equal to the value stored in the variable `find` what will the value returned by method `look` equal?

A. `vals.length` B. `0`

C. `(vals.length/2)` D. `1`

E. `-(vals.length/2)`

QUESTION 32

The following values are inserted one at a time into a binary search tree using the traditional insertion algorithm. What is the result of an in-order traversal of the resulting tree?

`5, 12, 0, -3, 9`

A. `-3 0 5 9 12` B. `5 12 0 -3 9` C. `12 9 5 0 -3`

D. `0 -3 5 9 12` E. `5 0 -3 9 12`

QUESTION 33

Given the following measurements, what is the most likely running time for method `sample(int[] data)` where N is equal to `data.length`? Choose the most restrictive correct answer.

Value of N	Time for method `sample` to complete
2,000	1 second
4,000	2 seconds
6,000	3 seconds

A. O(N) B. O(NlogN) C. O(N^2) D. O(1) E. O($N^{3/2}$)

QUESTION 34

What replaces **<*1>** in the code to the right to place the value stored in the variable `x` at the end of `data` if the Boolean expression `x % 2 == 0` is true?

I. `data.add(x)`
II. `data.addLast(x)`
III. `x = data.removeFirst()`

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

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

```
public void test(LinkedList<Integer> data,
                                    int x){
  if(x % 2 == 0){
    <*1>;
  }
}
```

QUESTION 35

Which sorting algorithm do the two methods to the right named `sort` implement?

A. merge sort

B. selection sort

C. bubble sort

D. quicksort

E. insertion sort

QUESTION 36

What is the Big O of the method named `sort` with a single parameter given an array of `int`s that is already sorted into ascending order? N = `data.length`. Choose the most restrictive correct answer.

A. O(N)

B. O(NlogN)

C. $O(N^{3/2})$

D. $O(N^2)$

E. $O(N^3)$

```
public void sort(int[] data){
  int[] temp = new int[data.length];
  sort(data, temp, 0, data.length - 1);
}

public void sort(int[] data,
                 int[] temp, int i, int j){
  if(i < j){
    int mid = (i + j) / 2;
    sort(data, temp, i, mid);
    sort(data, temp, mid + 1, j);

    int le = mid;
    int tp = i;
    int ne = j - i + 1;
    while( (i <= le) && (mid + 1 <= j) ){
      if( data[i] <= data[mid + 1] )
        temp[tp] = data[i++];
      else
        temp[tp] = data[mid++ + 1];
      tp++;
    }

    while( i <= le)
      temp[tp++] = data[i++];

    while( mid + 1 <= j)
      temp[tp++] = data[mid++ + 1];

    for(int k = 0; k < ne; k++){
      data[j] = temp[j];
      j--;
    }
  }
}
```

QUESTION 37

What is output by the code to the right?

A. `02468` B. `0` C. `10`

D. `0246810` E. `024`

```
Queue<Integer> q;
q = new LinkedList<Integer>();

for(int i = 0; i < 10; i += 2)
  q.add(i);

for(int i = 0; i < q.size(); i++)
  System.out.print( q.remove() );
```

QUESTION 38

What is output by the client code to the right?

A. `frums`

B. `fmrsu`

C. `usrmf`

D. `fffff`

E. `smurf`

QUESTION 39

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

A. A binary search tree

B. A stack

C. A priority queue

D. A queue

E. A linked list

```
public class Structure<E>{

  LinkedList<E> con;

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

  public void add(E obj){
    con.addFirst(obj);
  }

  public E access(){
    return con.getFirst();
  }

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

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

// client code
Structure<Character> st;
st = new Structure<Character>();
String cartoon = "smurf";

for(int i = 0; i < cartoon.length(); i++)
  st.add( cartoon.charAt(i) );

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

QUESTION 40

What is output when method `kick` is called if `mat` is the 2D array below?

1	4	8	-5	8
3	3	8	1	0
2	0	7	7	5
-4	4	3	3	3
0	2	0	4	1

A. `11000` B. `11111` C. `00000`

D. `00111` E. `00101`

```
public void kick(int[][] mat){
  for(int i = 0; i < mat.length; i++)
    System.out.print( off(mat, i) );
}

public int off(int[][] mat, int i){
  int r = 0;
  int c = 0;
  for(int j = 0; j < mat.length; j++){
    r += mat[i][j];
    c += mat[j][i];
  }

  return (r > c) ? 0 : 1;
}
```

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 1 2009

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

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.

22. `List` is an interface. Interfaces cannot be instantiated.

30. The expression is evaluated `vals.length + 1` times. It is true `vals.length` times and false once.

36. This version of merge sort is still O(NlogN) even if the data is already sorted.

37. The code does not remove all elements in the queue because the size of the queue is being reduced by the remove operation while the loop control variable is increasing.