University Interscholastic League

Computer Science Competition

Number 118 (State - 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 does 1101100_2 minus 1111_2 equal?

A. $5D_{16}$ B. 1100011_2 C. 103_{10} D. F_{16} E. 1001111_2

QUESTION 2

What is output by the code to the right?

A. 5.5 B. 6.5 C. 8.75

D. 7.5 E. 6.0

```
double a = 2.5;
int x = 3;
a = a * 2 + (x / 2);
System.out.println( a );
```

QUESTION 3

What is output by the code to the right?

A. 15 B. 11 C. 10

D. 5 E. 4

```
int hold = 15;
for(int i = 0; i <= 10; i++){
  hold--;
}
System.out.print( hold );
```

QUESTION 4

What is output by the code to the right?

A. 12 B. 9 C. 3

D. 6 E. 15

```
String t1 = "eiffel";
String t2 = "php";
String t3 = t2 + t1 + t2;
System.out.print( t3.length() );
```

QUESTION 5

What is output by the code to the right?

A. null B. -1 C. 0

D. 1 E. 3

```
int[] fibs = {1, 1, 2, 3, 5, 8, 13};
fibs[3] = fibs[0];
fibs[0]--;
System.out.print( fibs[3] );
```

QUESTION 6

What is output by the code to the right?

A. 2 B. 30 C. 10

D. 12 E. 0

```
int r = 20;
int s = 10;
int t = s + s + s + s / r;
System.out.print( t );
```

QUESTION 7

Which answer is logically equivalent to the following Boolean expression, where `x, y and z` are `int` variables?.

```
(x > y) && !(y <= z)
```

A. `!(x <= y) && (y > z)` B. `!(x > y) && !(y <= z)`

C. `(x > y) || !(y <= z)` D. `(x <= y) && !(y <= z)`

E. `!( (x > y) && !(y <= z) )`

QUESTION 8

What is output by the code to the right?

A. 3 B. 23 C. 13

D. 1 E. 2

```
String home = "";
if( home == null )
  System.out.print( 1 );
else
  System.out.print( 2 );
if( home.length() != 0 )
  System.out.print( 3 );
```

QUESTION 9

How many constructors does the `Table` class have?

A. 2

B. 0

C. 4

D. 1

E. 3

QUESTION 10

What is output by the client code to the right?

A. `null, legs: 0`

B. `, legs: 4`

C. `Ikea, legs: 4`

D. `blob, legs: 4`

E. `Ikea, legs: 0`

```
public class Furniture{

  private String name;

  public Furniture(){
    name = "blob";
  }

  public Furniture(String n){
    name = n;
  }

  public String toString(){
    return name;
  }
}

public class Table extends Furniture{
  private int legs;

  public Table(int n){
    super("Ikea");
    legs = n;
  }

  public String toString(){
    String result = super.toString();
    result += ", legs: " + legs;
    return result;
  }
}

////////////////////////////////////////
// client code
Table endTable = new Table(4);
System.out.print( endTable );
```

QUESTION 11

What is output by the code to the right?

A. -2 B. -1 C. 0

D. 1 E. -2147483648

```
int m = 1;
int n = ~m;
System.out.print( n );
```

QUESTION 12

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

A. Variable `b2` has not been initialized.

B. The `Math.round` method does not exist.

C. `long`s may not be stored in `int` variables without casting.

D. Arguments to the `Math.round` method cannot be less than 0.

E. `double`s may not be stored in `int` variables without casting.

```
double b2 = -2.55;
int x = Math.round( b2 );
```

QUESTION 13

What is output by the code to the right?

A.
```
A
GREAT
FILM
```

B. `A \`

C. `A "GREAT" film`

D. `A \"GREAT\" film`

E. `A GREAT FILM`

```
System.out.print("A \"GREAT\" film");
```

QUESTION 14

What is output by the code to the right?

A. `2009` B. `+2009` C. `002009`

D. `d+2009` E. `+02009`

```
System.out.printf("%+06d", 2009);
```

QUESTION 15

What is returned by the method call `happy( happy(2, 3), happy(3, 2) )`?

A. `10` B. `9` C. `21`

D. `7` E. `18`

```
public int happy(int x, int y){
  x--;
  ++y;
  return x * y;
}
```

QUESTION 16

What is output by the code to the right?

A. `0` B. `2` C. `3`

D. `4` E. `16`

```
int counter = 0;
String pos = "Dean_of_Students";

for(int i = 0; i < pos.length(); i++){
  char ch = pos.charAt(i);
  if( Character.isUpperCase( ch ) )
    counter++;
}

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

QUESTION 17

Which of the following best describes what the code to the right will output?

A. If `b` is initialized to `true` the code prints `true`, otherwise it prints `false`.

B. If `b` is initialized to `true` the code prints `false`, otherwise it prints `true`.

C. The code always prints out `truefalse`.

D. The code always prints out `true`.

E. The code always prints out `false`.

```
boolean b;
// code to initialize b;

boolean oldB = b;
b = (b == false);
System.out.print( b == oldB );
```

QUESTION 18

What is output by the code to the right?

A. `falsefalse` B. `falsetrue`

C. `truefalse` D. `truetrue`

E. `true`

```
ArrayList<String> f;
f = new ArrayList<String>();
List<String> s;
s = new LinkedList<String>();

System.out.print(f instanceof List);
System.out.print(s instanceof LinkedList);
```

QUESTION 19

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

A. `b3a63` B. `3db63` C. `3ab66`

D. `b3a66` E. `3ab63`

```
public int pi(int x){
  System.out.print( x + "a" );
  return x * 2;
}

public void rho(){
  int y = 3;
  System.out.print( "b" + pi(y) + y );
}
```

QUESTION 20

What is returned by method `enigma` if `data` is the array shown below?

`{2, 0, 1, 3, -5, 2, 5, -3}`

A. `9` B. `4` C. `8`

D. `-1` E. `5`

```
public int enigma(int[] data){
  int i = 0;
  for(; i < data.length; i++){
    if( data[i] < 0 )
      break;
  }
  return i == data.length ? -1 : i;
}
```

QUESTION 21

What is output by the code to the right?

A. `2` B. `3` C. `4`

D. `5` E. `6`

```
HashSet<String> set;
set = new HashSet<String>();

set.add("A");
set.add("B");
set.add("AA");
set.add("B");

System.out.print( set.size() );
```

QUESTION 22

Which of the following can replace **<*1>** in the code to the right so that the code segment compiles without error?

I. `Collection<Integer>`
II. `Object`
III. `Queue<Integer>`

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

D. I, II, and III E. None of the choices.

```
<*1> tally = new LinkedList<Integer>();
```

QUESTION 23

What is output by the code to the right?

A. `0`

B. `10`

C. `45`

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

E. There is no output due to an infinite loop that occurs when the code is run.

```
ArrayList<Integer> nums;
nums = new ArrayList<Integer>();
for(int i = 0; i < 10; i++)
  nums.add(i);

Iterator<Integer> it = nums.iterator();
int count = 0;
while( it.hasNext() )
  count++;

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

QUESTION 24

What can replace **<*1>** in the code to the right so that the code segment compiles without error.

A. Any valid identifier that is not already in scope.

B. Only the identifier `e`.

C. Only the identifier `this.Exception`.

D. Any single digit.

E. One or more `&`'s.

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

QUESTION 25

What is output by the code to the right?

A. `e2`

B. `454`

C. `e297`

D. `e1`

E. `e1e2`

```
try{
  int[] passnums = {31, 2, 45, 4, 97};
  int i1 = passnums[2];
  int i2 = passnums[3];

  System.out.print( passnums[i1] );
  System.out.print( passnums[i2] );
}
catch(NullPointerException <*1>){
  System.out.print( "e1" );
}
catch(ArrayIndexOutOfBoundsException <*1>){
  System.out.print( "e2" );
}
```

QUESTION 26

What is output by the code to the right?

A. ULBDER B. EULB

C. EEEEEEE D. REDBLUE

E. EULBDER

```
String colors = "REDBLUE";
Stack<Character> st;
st = new Stack<Character>();

for(int i = 0; i < colors.length(); i++)
  st.push( colors.charAt(i) );

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

QUESTION 27

Which of the following can replace **<*1>** in the code to the right so that method `passItOn` compiles without error?

I. `LinkedList<Integer>`
II. `HashSet<String>`
III. `ArrayList<Map.Entry<String, Integer>>`

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

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

```
public void passItOn(<*1> coll){
  Collections.sort( coll );
}
```

QUESTION 28

What Java programming language feature allows the primitive `int`s to be used as arguments to the constructor calls in the client code to the right even though the data type of the parameter `d` is `Object`, not `int`?

A. exceptions

B. static variables

C. autoboxing

D. recursion

E. method overloading

```
public class Node{
  public Node one;
  public Node two;
  public Object data;

  public Node(Node o, Node t, Object d){
    one = o;
    two = t;
    data = d;
  }

  public Node(){
  }

}

////////////////////////////////////////
// client code
Node n1 = new Node(null, null, 1);
Node n2 = new Node(n1, new Node(), 2);
Node n3 = new Node(n1, n2, 3);

n1.two = n3;
n2.two.one = n3.one;
n1.one = n3.two.one;

System.out.println( n2.one.two.one.data );
```

QUESTION 29

What is output by the client code to the right?

A. 1

B. 2

C. 3

D. There is no output due to a `ArrayIndexOutOfBoundsException`.

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

QUESTION 30

What is output by the code to the right?

A. 1 B. 3 C. -1

D. -19 E. -3

```
String s1 = "CAN";
String s2 = "CANTOR";
System.out.print( s1.compareTo( s2 ) );
```

QUESTION 31

If N equals `d.size()` what is the average running time of method `search` when `d` is an `ArrayList` and when `d` is a `LinkedList`? Pick the most restrictive correct set of answers.

	ArrayList	LinkedList
A.	O(N)	O(logN)
B.	O(logN)	O(NlogN)
C.	O(NlogN)	$O(N^2)$
D.	O(logN)	O(logN)
E.	O(l)	O(logN)

QUESTION 32

What is output by the client code to the right?

A. `0`

B. `1000`

C. `499`

D. `999`

E. `500`

```
// pre: list != null and
// elements in list are sorted in
// ascending order.
public int search(List<Integer> d,
                                    int tgt){
  Integer t = new Integer(tgt);
  int res = -1;
  int low = 0;
  int hi = d.size() - 1;
  int count = 0;
  while( res == -1 && low <= hi ){
    count++;
    int mid = (low + hi) / 2;
    int diff = t.compareTo( d.get(mid) );
    if( diff == 0 )
      res = mid;
    else if( diff > 0 )
      low = mid + 1;
    else
      hi = mid -1;
  }
  return res;
}

////////////////////////////////////////
// client code
ArrayList<Integer> t;
t = new ArrayList<Integer>();
for(int i = 0; i < 1000; i++)
  t.add( 0 );
System.out.print( search(t, 0) );
```

QUESTION 33

Assume method `fib(int[] data)` is $O(2^N)$ where N = `data.length`. When method `fib` is passed an array with `length` = 50 it takes 0.5 seconds for method `fib` to complete. If method `fib` is then passed an array with `length` = 54 what is the expected time it will take method `fib` to complete?

A. 0.54 seconds B. 4 seconds C. 16 billion seconds D. 8 seconds E. 0.51 seconds

QUESTION 34

What is output when method `work` is called if `data` is the array shown below?

`{3, 2, 3, 0, 4, 0, 3, 1, 5, 0}`

A. `1`

B. `21`

C. `15`

D. `0`

E. `1080`

```
public void work(int[] data){
  int result = 1;
  for(int i = 0; i < data.length; i++){
    if( data[i] != 0 )
      result *= data[i];
    else
      result = 1;
  }
  System.out.print(result);
}
```

QUESTION 35

The `Arrays.sort(int[] a)` method calls a helper method with the following header:

```
private static void sort1(int x[], int off, int len) {
```

The parameters `off` and `len` specify a sub-array in `x` to be sorted. `len` is the length of the sub-array.
The implementation of the method `sort1` is:

```
if( len < 7 )
    // perform an insertion sort on the sub-array
else
    // perform a quicksort on the sub-array
```

Which of the following is the best reason the `sort1` method uses this hybrid (combination of quicksort and insertion sort) sorting algorithm?

A. So that the sort will work on all primitive integer types: `byte`, `short`, `int` and `long`.

B. Primitive `int`s do not have a `compareTo` method.

C. So that the sort will be stable, meaning the relative order of equal items in the original array is unchanged.

D. So that an auxiliary linked list is not needed to complete the sort.

E. It is usually faster to sort a small array using the insertion sort algorithm rather than the quicksort algorithm.

QUESTION 36

What is output by the statement marked
`// line 1` in the client code to the right?

A. `0`

B. `1`

C. `5`

D. `16`

E. `32`

QUESTION 37

What is output by the statement marked
`// line 2` in the client code to the right?

A. `[2, 3]`

B. `[]`

C. `[2, 3, 3]`

D. `8`

E. `[8, 5]`

```
public void ps(int[] d, int p,
       ArrayList<Integer> cur,
       ArrayList<ArrayList<Integer>> res){

  if(p == d.length)
    res.add( gc(cur) );
  else{
    ps(d, p + 1, cur, res);
    cur.add(d[p]);
    ps(d, p + 1, cur, res);
    cur.remove( cur.size() - 1 );
  }
}

public ArrayList<Integer> gc(
                        ArrayList<Integer> org){
  ArrayList<Integer> r;
  r = new ArrayList<Integer>();
  for(int i : org)
    r.add(i);
  return r;
}

//////////////////////////////////////////
// client code
int[] ds = {2, 3, 8, 3, 5};
ArrayList<ArrayList<Integer>> res;
res = new ArrayList<ArrayList<Integer>>();
ArrayList<Integer> c = new ArrayList<Integer>();
ps(ds, 0, c, res);

System.out.println( res.size() ); // line 1
System.out.println( res.get(5) ); // line 2
```

QUESTION 38

What replaces **<*1>** in the code to the right to decrement the value stored inside the variable `p`?

A. `p -= 1`

B. `p *= -1`

C. `p >> 2`

D. `p << 1`

E. `p++`

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

QUESTION 39

What is output by the client code to the right?

A. `MM`

B. `UU GG MM GG`

C. `GG`

D. `12`

E. `UU`

QUESTION 40

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

A. a stack

B. a heap

C. a list

D. a binary search tree

E. a priority queue

```
public class Structure{

  private ArrayList<Integer> ks;
  private ArrayList<Object> vs;

  public Structure(){
    ks = new ArrayList<Integer>();
    vs = new ArrayList<Object>();
  }

  public void add(int k, Object v){
    int p = ks.size();
    while(p > 0 && k <= ks.get(p - 1) )
      <*1>;
    ks.add(p, k);
    vs.add(p, v);
  }

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

  public Object access(){
    return vs.get( ks.size() - 1 );
  }

  public Object remove(){
    ks.remove(ks.size() - 1);
    return vs.remove( vs.size() - 1 );
  }
}

////////////////////////////////////////
// client code

Structure str = new Structure();

str.add(10, "GG");
str.add(12, "MM");
str.add(12, "GG");
str.add(5, "UU");

System.out.print( str.access() );
```

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 State 2009

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

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.

9. Constructors are not inherited and if any constructors are declared the implicit default constructor is lost.

11. The `~` operator flips all the bits in the expression. So 00000000 00000000 00000000 00000001 becomes 11111111 11111111 11111111 11111110 which is the 32 bit two's complement representation of -2.

23. The `iterator` method `next()` is never called on the non empty list, resulting in an infinite loop.

27. The `Collections.sort` method requires the argument to be a class that implements the `List` interface and that stores a type that implements the `Comparable` interface. The method header is:
`public static <T extends Comparable<? super T>> void sort(List<T> list)`

30. For two `String`s with different lengths and where one is a prefix of the other he `String compareTo` method returns the difference of the lengths: `this.length() - anotherString.length()`

31. The `get` method from the `LinkedList` class is O(N) and it is called $\log_2 N$ times giving a running time of O(NlogN) when `d` is a `LinkedList`.