University Interscholastic League

Computer Science Competition

Number 117 (Regional - 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 $1BC_{16}$ and 11100111_2?

A. FBC_{16} B. $2A3_{16}$ C. 717_{10} D. 679_8 E. 369_{16}

QUESTION 2

What is output by the code to the right?

A. 18 B. 15 C. 8

D. 335 E. 14

```
int x = 3;
int y = x + 1 * x + 2;
System.out.println( y );
```

QUESTION 3

What is output by the code to the right?

A. 5 12 B. 4 8 C. 6 10

D. 5 10 E. 6 12

```
int store = 0;
int i;
for(i = 1; i <= 5; i++){
  store += 2;
}
System.out.print( i + " " + store );
```

QUESTION 4

What is output by the code to the right?

A. c==c3basic B. c++c#basic

C. false D. ccbasic

E. C++C#BASIC

```
String langs = "C++C#BASIC";
System.out.print( langs.toLowerCase() );
```

QUESTION 5

What is output by the code to the right?

A. 1 B. 2 C. 12

D. 21 E. The code to the right does not produce any output.

```
boolean[] ans = new boolean[5];
char[] lets = new char[5];

if( ans[2] )
  System.out.print( "1" );
if( lets[0] == '0' )
  System.out.print( "2" );
```

QUESTION 6

What is output by the code to the right?

A. 34 B. 3 C. 10

D. 12 E. 38

```
int r = 31;
int s = 7;
System.out.print( r % s + s % r );
```

QUESTION 7

How many of the 8 possible combinations of values for the variables `a`, `b`, and `c` will result in `d` being set to `true`?

A. 3 B. 1 C. 0

D. 5 E. 8

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

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

QUESTION 8

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

I. `0`
II. `1`
III. `2`

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

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

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

QUESTION 9

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

A. `3`

B. `4`

C. `2`

D. `1`

E. `0`

QUESTION 10

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

A. `Mon_Mon`

B. `Mon_3`

C. `Mon_2`

D. `1_3`

E. `Mon_0`

```
public class Timestamp{

  private static int myst = 0;

  private String day;
  private int hour;

  public Timestamp(String d){
    this(d, 0);
    myst++;
  }

  public Timestamp(String d, int h){
    day = d;
    hour = h;
    myst++;
  }

  public static int getMyst(){
    return myst;
  }

  public String toString(){
    return day + "_" + hour;
  }

  public static void reset(){
    myst = 0;
  }
}

////////////////////////////////////////
// client code
Timestamp.reset();
Timestamp t1 = new Timestamp("Sun");
Timestamp t2 = new Timestamp("Mon", 3);

int tot = Timestamp.getMyst();
System.out.println( tot ); // line 1
System.out.println( t2 ); // line 2
```

QUESTION 11

What is output by the code to the right?

A. `0` B. `256` C. `640`

D. `true` E. `1010000000`

```
int m = 512;
int n = 128;
System.out.print( n | m );
```

QUESTION 12

What is output by the code to the right?

A. `1.0` B. `0` C. `0.0`

D. `2.0` E. `1`

```
double a2 = 1.05;
System.out.print( Math.ceil(a2) );
```

QUESTION 13

What is output by the code to the right?

A. `105=2` B. `10\\5=2` C. `10\\\5=2`

D. `10\5=2` E. `10\\\\5=2`

```
String prob = "10\\\\5=";
int ans = 2;
System.out.print( prob + ans );
```

QUESTION 14

What is output by the code to the right?

A. `0.315` B. `.315` C. `(0.315)`

D. `-(0.315)` E. `.3150`

```
System.out.printf("%(4.3f", .315);
```

QUESTION 15

What is returned by the method call `adjust(1.7)`?

A. `3.0` B. `1.7` C. `3.6`

D. `4.0` E. `5.4`

```
public double adjust(double a){
  a++;
  a *= 2;
  return a;
}
```

QUESTION 16

What is output by the code to the right?

A. `3sc12` B. `3sc3` C. `12sc12`

D. `sc` E. `1+2+mid+1+2`

```
String mid = "sc";
String result = 1 + 2 + mid + 1 + 2;
System.out.print( result );
```

QUESTION 17

What is output by the code to the right?

A. `12` B. `8` C. `9.0`

D. `18` E. `9`

```
int z = 4;
double a = 2.5;
System.out.print( (int) a * 2 + z );
```

QUESTION 18

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

A. `null`

B. `5`

C. `5null`

D. There is no output.

E. The output cannot be determined until runtime.

QUESTION 19

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

A. `10`

B. `10gack`

C. `gack`

D. `5gack`

E. `gack5`

```
public class Problem{
  private int points;

  public Problem(int pts){
    points = pts;
  }

  public int getPoints(){
    return points;
  }
}

public class HardProblem extends Problem{
  private String exclaim;

  public HardProblem(int pts, String e){
    super(pts);
    exclaim = e;
  }

  public String toString(){
    return super.getPoints() + exclaim;
  }
}

////////////////////////////////////////
// client code
Problem p = new Problem(5);
HardProblem hp;
hp = new HardProblem(10, "gack");

System.out.println( p ); // line 1
System.out.println( hp ); // line 2
```

QUESTION 20

What is output by the line of code to the right that comes after the comment `// Question 20`?

A. `2` B. `4` C. `7`

D. `4,7` E. `2,2`

QUESTION 21

What is output by the code segment to the right that comes after the comment `// Question 21`?

A. `SANTANA`

B. `ANST`

C. `SANT`

D. `S1A3N2T1`

E. The output cannot be determined until runtime.

```
TreeSet<String> ts = new TreeSet<String>();
String lets = "SANTANA";

for(int i = 0; i < lets.length(); i++)
  ts.add( lets.charAt(i) + "" );

// Question 20
System.out.println( ts.size() );

// Question 21
for(String s : ts)
  System.out.print( s );
```

QUESTION 22

Which of the following statements about classes that have the clause `implements Collection` in their class header are true?

I. The class can never store duplicate elements.
II. The class can never be declared `abstract`.
III. The elements stored in the class must always be kept in order based on the `compareTo` method.

A. I only B. II only C. III only D. I and III E. None of the statements are true.

QUESTION 23

What is output by the code to the right?

A. `08` B. `10` C. `09`

D. `9` E. `5`

```
String times;
times = "7:48:1:09:08";
String[] pieces = times.split(":");
System.out.print( pieces[3] );
```

QUESTION 24

What replaces **<*1>** in the code to the right to set the variable `cs` equal to the number of columns in the two-dimensional array of `int`s named `t`?

A. `t[0].length` B. `t.0.length`

C. `t->length` D. `t.length`

E. `t[0][0].length`

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

QUESTION 25

What is returned by method `handle` if `t` is the matrix shown below?

1	4	0	2	1	6
0	-1	5	4	0	-4
2	2	7	1	13	2
11	5	13	13	4	20

A. `15` B. `18` C. `19`

D. `14` E. `16`

```
public int handle(int[][] t){
  int tot = 0;
  int rs = t.length;
  int cs = <*1>;
  int start = Math.min(rs, cs) - 1;
  int m = start / 2;
  for(int i = start; i >= 0; i--){
    tot += t[i][m];
    tot += t[m][i];
  }
  return tot;
}
```

QUESTION 26

What is output by the code to the right?

A. `1.100` B. `1100.0` C. `1.331`

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

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

```
String value = "1.1e3";
double a = Double.parseDouble( value );
System.out.print( a );
```

QUESTION 27

What replaces **`<*1>`** in the code to the right to indicate method `count` will pass along any `FileNotFoundExceptions` it may generate instead of handling them locally?

A. `throws FileNotFoundException`

B. `finally FileNotFoundException`

C. `throws new FileNotFoundException()`

D. `throw new FileNotFoundException`

E. `catch FileNotFoundException`

```
public int count(String filename) <*1> {
  Scanner sc;

  // The next line of code can result in
  // a FileNotFoundException.
  sc = new Scanner( new File(filename) );

  int count = 0;

  // rest of method not shown

  return count;
}
```

QUESTION 28

What is output by the code to the right?

A. `2` B. `4` C. `6`

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

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

```
int[] sample = {2, 4, 6};
Iterator<Integer> it = sample.iterator();
it.next();
it.next();
System.out.println( it.next() );
```

QUESTION 29

Which of the following best describes what method `mystery` does if the precondition that `text` does not equal `null` is met?

A. Always returns the number of spaces in `text`.

B. Always returns the number of tokens in `text`.

C. Always returns the number of tokens in `text` that start with the character `'A'`.

D. Prints out all the tokens in `text` that start with the character `'A'`.

E. Always returns the number of `'A'`s in `text`.

```
// pre: text != null
public int mystery(String text){
  Scanner sc = new Scanner(text);
  int count = 0;
  while( sc.hasNext() ){
    String temp = sc.next();
    int len = temp.length();
    if( len > 0 && temp.charAt(0) == 'A' )
      count++;
  }
  return count;
}
```

QUESTION 30

Assume method `performAction(int[] data)` is $O(N^2)$ where N = `data.length`. When method `performAction` is passed an array with `length` = 2,000 it takes 4 seconds for method `performAction` to complete. If method `performAction` is then passed an array with `length` = 4,000 what is the expected time it will take the method to complete?

A. 8 seconds B. 16 seconds C. 28 seconds D. 32 seconds E. 64 seconds

QUESTION 31

What replaces **<*1>** in the code to the right so that the body of the `if` statement is executed if one or more of the conditions `p == vs.length` and `cap == 0` evaluate to `true`?

A. `^` B. `||` C. `^^`

D. `%%` E. `&&`

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

```
public int find(int[] vs, int[] ws,
                  int p, int val, int cap){

  if( p == vs.length <*1> cap == 0 )
    return val;

  int wo = find(vs, ws, p + 1, val, cap);
  int th = 0;
  if( ws[p]<= cap)
    th = find(vs, ws, p + 1,
                  val + vs[p], cap - ws[p]);
  return Math.max(wo,th);
}

// client code
int[] vs = {6, 5, 9, 6, 2};
int[] ws = {4, 4, 8, 4, 2};
System.out.println(find(vs, ws, 0, 0, 10));
```

QUESTION 32

What is output by the client code to the right?

A. `13` B. `14` C. `8`

D. `11` E. `28`

QUESTION 33

Which of the following can replace **<*1>** in the code to the right to always set the variable `z` to the minimum of the variables `x` and `y`?

I. `Math.min(x,y)`
II. `(x < y) ? x : y`
III. `x && y`

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

D. I and II E. II and III

```
int x;
int y;
// code to initialize x and y

int z = <*1>;
```

QUESTION 34

A sort is defined to be *stable* for a given array if equal elements in the original array maintain their relative positions in the sorted array. For example consider the following array of `int`s:

`{0, 7, 5, 3, 7}`

If the sort is stable for this array then in the sorted array, the `7` originally at index 1 will be before the `7` originally at index 4.

Method `sort` to the right implements the selection sort algorithm. For what input arrays is method `sort` stable?

A. All arrays

B. Some arrays

C. No arrays

D. It is not possible to determine if the method `sort` is stable or not.

E. More than one of these is correct.

```
public void sort(int[] vals){
  int minIndex;
  int limit = vals.length;
  for(int i = 0; i < limit ; i++){
    minIndex = i;
    for(int j = i + 1; j < limit ; j++){
      if( vals[j] < vals[minIndex] ){
        minIndex = j;
      }
    }
    int temp = vals[i];
    vals[i] = vals[minIndex];
    vals[minIndex] = temp;
  }
}
```

QUESTION 35

Method `search` to the right implements the binary search algorithm. If `list.length` is 256 what is the largest possible `value` the method will print out at the line of code marked `// line 1`?

A. `1`

B. `6`

C. `9`

D. `25`

E. `257`

```
// pre: list != null and
// elements in list are sorted in
// ascending order.
public int search(int[] list, int tgt){
  int res = -1;
  int low = 0;
  int hi = list.length - 1;
  int count = 0;
  while( res == -1 && low <= hi ){
    count++;
    int mid = (low + hi) / 2;
    if( list[mid] == tgt )
      res = mid;
    else if( list[mid] < tgt )
      low = mid + 1;
    else
      hi = mid -1;
  }
  System.out.println( count ); // line 1
  return res;
}
```

QUESTION 36

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

A. `new HashSet()`

B. `new HashMap(Character, Integer)`

C. `new HashMap<Character, Integer>()`

D. `new Map<Character, Integer>()`

E. More than one of these are correct.

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

QUESTION 37

What is output by the code to the right?

A. `1` B. `3`

C. `4` D. `7`

E. The output cannot be determined until runtime.

```
String word = "riffraf";
Map<Character, Integer> tags;
tags = <*1>;
for(int i = 0; i < word.length(); i++){
  char ch = word.charAt(i);
  if( !tags.containsKey(ch) )
    tags.put( ch, 1 );
  else
    tags.put( ch, tags.get(ch) + 1 );
}
System.out.println( tags.size() );
```

QUESTION 38

What is output by the code to the right?

A. `null` B. `true` C. `false`

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

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

```
String[] langs = new String[10];
boolean isObject;
isObject = langs[2] instanceof Object;
System.out.print( isObject );
```

QUESTION 39

Consider the `Structure` class to the right. What is output by the following client code?

```
Structure s = new Structure();
s.add("dog");
s.add("dad");
s.add("cab");
s.add("add");
s.add("dad");
s.showAll();
```

A. `dog dad cab add dad`

B. `dog dad cab`

C. `add cab dad dog`

D. `cab add dad dog`

E. `cab dad add dog`

QUESTION 40

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

A. A binary search tree

B. A hash table

C. A stack

D. A heap

E. A queue

```
public class Structure{
  private LinkedList[] con;

  public Structure(){
    con = new LinkedList[100];
    for(int i = 0; i < con.length; i++)
      con[i] = new LinkedList();
  }

  public void add(String obj){
    int val = getValue(obj);
    if( !con[val].contains(obj) )
      con[val].add(obj);
  }

  public boolean isPresent(String obj){
    int val = getValue(obj);
    return con[val].contains(obj);
  }

  public boolean remove(String obj){
    int val = getValue(obj);
    return con[val].remove(obj);
  }

  public void showAll(){
    for( LinkedList<String> i : con )
      for( String st : i )
        System.out.print( st + " " );
  }


  private int getValue(String obj){
    int val = 0;
    obj = obj.toLowerCase();
    for(int i = 0; i < obj.length(); i++){
      char ch = obj.charAt(i);
      if( Character.isLetter( ch ) )
        val += ch - 'a';
    }
    return val % con.length;
  }
}
```

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 Regional 2009

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

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.

7. The `&&` which is the *logical and* operator has a higher precedence than `||` which is the *logical or* operator. This causes the expression `b && c` to be evaluated first. In other words the expression is equivalent to `( a || (b && c) )`.

9. The static variable `myst` will equal 3 at this point. The first constructor call in the client code results in `myst` being incremented twice.

18. The `Problem` class does not override `Object`'s `toString` method which prints out the hashcode for the object. The hashcode cannot be determined until the code is actually run and may vary from one run to the next.

28. Although native arrays are *`Iterable`* meaning they can be the target of a "foreach" loop the language does not allow explicit creation of an iterator object for an array.

Notes continued on back.

34. B Sometimes. An example of when the sort is stable is the initial array `{1, 2, 1}`. An example of when the sort is not stable is the initial array `{2, 2, 1}`.

35. The body of the while loop will be executed 9 times when searching for a value larger than the maximum value in `list`.

38. The expression `null instanceof Object` always evaluates to false.