University Interscholastic League

Computer Science Competition

Number 106 (State - 2007)

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. You may 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 11100101_2 minus BF_{16} equal?

A. 40_8 B. $D6_{16}$ C. 306_8 D. 46_8 E. 1001110_2

QUESTION 2

What is output by the code to the right?

A. `18-4` B. `24-4` C. `24-24`

D. `14-4` E. `14-14`

```
int x = 3;
int y = x;
y++;
x *= y + 2;
System.out.print( x + "-" + y );
```

QUESTION 3

What is output by the code to the right?

A. `120` B. `124` C. `129`

D. `144` E. `15`

```
int total = 0;
for(int i = 0; i < 4; i++){
    total++;
    for(int j = 0; j < 5; j++){
      total++;
      for(int k = 0; k < 6; k++)
        total++;
    }
}
System.out.print( total );
```

QUESTION 4

What is output by the code to the right?

A. `tboy` B. `ertboy` C. `rtboyer`

D. `ertboye` E. Nothing is printed out.

```
String s1 = "robertboyer";
int p1 = s1.indexOf("er");
int p2 = s1.indexOf("er", p1 + 1);
s1 = s1.substring(p1, p2);
System.out.println( s1 );
```

QUESTION 5

What is output by the code to the right?

A. `2-21`

B. `7-15`

C. `1-5`

D. `0-7`

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

```
int[] list = {7, 2, 20, 14, 6, 21, 15};
int m1 = 0;
int m2 = 0;

for(int i = 1; i < list.length - 1; i++){
  if( list[i] < list[i + 1] ){
    if( list[i] < list[m1] )
      m1 = i;
    if( list[i+1] > list[m2] )
      m2 = i;
  } else {
    if( list[i + 1] < list[m1] )
      m1 = i + 1;
    if( list[i] > list[m2] )
      m2 = i;
  }
}

System.out.print( m1 + "-" + m2 );
```

QUESTION 6

What is output by the code to the right?

A. `0149` B. `481632`

C. `491625` D. `251694`

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

```
int[][] mat = new int[4][4];
for(int i = 0; i < 4; i++)
  for(int j = 0; j < 4; j++)
    mat[j][i] = (i + j) * (i + j);

for(int k = 0; k < 4; k++)
    System.out.print( mat[k][2] );
```

QUESTION 7

Which of the following replaces **`<*1>`** in the code to the right to set the variable `b` to `true`?

A. `r.toLowerCase().equals(s)`

B. `(r + t).equals(s)`

C. `Integer.parseInt(t) > s.length()`

D. `s.compareTo(r) == 0`

E. More than one of these.

```
boolean b;
String r = "abc";
String s = "ABC5";
String t = "5";
b = <*1>;
```

QUESTION 8

What is output by the code to the right?

A. `1` B. `2` C. `3`

D. `12` E. `13`

```
double a = 7.5;
double b = a / 2;
if( a > b * 2 )
    System.out.print(1);
if( b / a > 3 || a * b > 3 )
    System.out.print(2);
else
    System.out.print(3);
```

QUESTION 9

The code to the right results in the following output:

```
false
```

Which of these best explains why?

A. The `==` operator compares references to objects, not the objects themselves.

B. The `Strings` `n2` and `n4` have differences in capitalization.

C. The `==` operator compares which methods have been called on an object.

D. The `Strings` `n2` and `n4` are not made up of the same characters in the same order.

E. The `==` operator compares variable names.

```
String n1 = "Computer";
String n2 = n1.substring(1,4);
String n3 = "OMP";
String n4 = n3.toLowerCase();
System.out.println( n2 == n4 );
```

QUESTION 10

What is output by the code to the right?

A. `_week_n`

B. `l_week_n`

C. `l_w`

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

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

```
String q1 =
          "real_week_neon_at_the_seaside.";

String[] chop = q1.split("e[aiou].");
System.out.print( chop[1] );
```

QUESTION 11

Which of these data types replaces **<*1>** in the code to the right so the parameter `other` is of the correct type to implement the `compareTo` method from the `Comparable` interface.

A. `T`

B. `Score`

C. `Object`

D. `Comparable<T>`

E. `Comparable`

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

QUESTION 12

Which of these replaces **<*2>** in the code to the right to correctly implement the `compareTo` method from the `Comparable` interface? A `Score` object is greater than another `Score` object if the value stored in its `pts` field is greater than the value stored in the other `Score` object's `pts` field.

A. `return pts - other.pts`

B. `return getPts() - other.getPts()`

C. `return other.pts - pts`

D. `return other.getPts() - getPts()`

E. More than one of these.

Assume **<*1>** and **<*2>** are filled in correctly.

QUESTION 13

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

A. 3 B. 4 C. 5

D. 6 E. 8

QUESTION 14

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

A. 1234

B. 4321

C. The output cannot be predicted.

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

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

```
public class Score
              implements Comparable<Score>{

  private int pts;

  public Score(int p){
    pts = p;
  }

  public void freeThrow(){
    pts++;
  }

  public void basket(){
    pts += 2;
  }

  public void trey(){
    pts += 3;
  }

  public int getPts(){
    return pts;
  }

  public int compareTo(<*1> other){
    <*2>;
  }
}

-------------------------------------------

// methods in a client of Score
public void mavs(){
  Score s1 = new Score(0);
  Score s2 = new Score(0);
  s1.basket();
  s2.trey();
  s1 = s2;
  s1.basket();
  s2.freeThrow();
  System.out.print( s1.getPts() );
}

public void spurs(){
  Set<Score> coll = new HashSet<Score>();
  for(int i = 1; i < 5; i++)
    coll.add( new Score(i) );
  for( Score s : coll )
    System.out.print( s.getPts() );
}
```

QUESTION 15

What is returned by the following method call?

`ways(new int[]{5, 3, 6, 3}, 15, 0 )`

A. `270` B. `17` C. `4`

D. `10` E. `18`

```
public static int ways(int[] ds,
                              int g, int c){
  int result = 0;
  if( c == ds.length ){
    if( g == 0 )
      result = 1;
  } else {
    for(int i = 1; i <= ds[c]; i++)
      result += ways(ds, g - i, c + 1);
  }
  return result;
}
```

QUESTION 16

What is the running time of method `manip`? Choose the most restrictive correct answer.

A. `O(1)` B. `O(n)` C. `O(nlogn)`

D. `O(n`^{2}`)` E. `O(n`^{2}`logn)`

```
public int manip(int n){
  int total = 0;
  for(int i = n; i > 0; i /= 2)
    for(int j = 0; j < i; j++)
      total += (j * i) % n;
  return total;
}
```

QUESTION 17

What is output by the code to the right?

A. `7-4.4` B. `7-2.2` C. `8-2.2`

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

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

```
int x3 = 3;
double a3 = 2.2;
x3 += a3 * 2;
System.out.print( x3 + "-" + a3 );
```

QUESTION 18

What is output by the code to the right?

A. `%1$04d,%2$+5.3f,365,4.1356`

B. `5554.136`

C. `0365,4.135`

D. `0365+4.1`

E. `0365,+4.136`

```
double planck = 4.1356;
int days = 365;
String format = "%1$04d,%2$+4.3f";
System.out.printf(format, days, planck);
```

QUESTION 19

What is output by the code to the right?

A. `253`

B. `-2`

C. `3`

D. `-3`

E. `-125`

```
byte val1 = 2;
byte val2 = (byte)~val1;
System.out.print( val2 );
```

QUESTION 20

What replaces **`<*1>`** in the code to the right to set `mid` to the middle element of the portion of the array `vals` from `st` to `fin`?

A. `mid = st + fin / 2`

B. `mid = st / 2 + fin`

C. `mid = 1.0 * (st + fin) / 2`

D. `mid = (st + fin) / 2`

E. More than one of these.

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

QUESTION 21

What is returned by the method call `demo1()` ?

A. `8` B. `0`

C. `9` D. `10`

E. There is no return value due to a runtime error.

QUESTION 22

What is returned by the method call `demo2()` ?

A. `0` B. `-1`

C. `4` D. `5`

E. There is no return value due to a runtime error.

QUESTION 23

If `vals` has a length of N and every element of `vals` equals the same value `x`, what is the running time of method `find(vals, x)`? Choose the most restrictive correct answer.

A. `O(x)`

B. `O(logN)`

C. `O(N)`

D. `O(N`^{2}`)`

E. `O(xN)`

```
// pre: elements in vals are sorted in
// ascending order
public static int find(int[] vals, int tgt){
  int st = 0;
  int fin = vals.length - 1;
  int mid;
  int result = -1;

  while( result == -1 && st <= fin ){
    <*1>;
    if( vals[mid] == tgt )
      result = mid;
    else if( vals[mid] < tgt )
      st = mid + 1;
    else
      fin = mid - 1;
  }

  while( result > 0
            && vals[ result - 1 ] == tgt ){
    result--;
  }

  return result;
}

public static int demo1(){
  int[] list = new int[20];
  int pos = 0;
  for(int i = -2; i <= 2; i++){
    for(int j = 0; j < 4; j++){
      list[pos] = i;
      pos++;
    }
  }

  return find( list, 0 );
}

public static int demo2(){
  int[] list = new int[10];
  for(int i = 0; i < 100; i += 10)
    list[ i / 10 ] = i;

  return find( list, 45 );
}
```

QUESTION 24

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

A. 4

B. 0

C. 3

D. 7

E. 5

```
public class IntCell{

  public int val;
  public IntCell next;

  public IntCell(int v){
    val = v;
  }

}
-------------------------------------------

// methods in a client of IntCell
public boolean one(IntCell x, IntCell y){
  x.val++;
  y.val += 2;
  return x.val < y.val;
}

public boolean two(IntCell x, IntCell y){
  x.val++;
  y.val--;
  return y.val <= x.val;
}

public void three(){
  IntCell a = new IntCell(6);
  IntCell b = new IntCell(3);
  if( one(a,b) && two(a,b) )
    System.out.print( a.val );
  else
    System.out.print( b.val );
}

public void four(){
  IntCell a = new IntCell(1);
  IntCell b = new IntCell(2);
  a.next = b;
  b.next = a.next;
  System.out.print( a.next.next.val );
}
```

QUESTION 25

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

A. 0

B. 1

C. 2

D. null

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

QUESTION 26

What is output by the code to the right?

A. 12_3

B. 8_3

C. 8_2

D. 12_2

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

```
int x = 3;
int y = 4;
int z = y * x--;
System.out.print( z + "_" + x );
```

QUESTION 27

What is output by the following client code?

```
Structure s1 = new Structure();
System.out.print( s1.findP(13) );
```

A. `1` B. `1000` C. `1011`

D. `101` E. `1101`

QUESTION 28

What is output by the following client code?

```
Structure s2 = new Structure();
int[] ents = {12, 5, 13, 17, -5};
for( int i : ents )
  s2.add( i );
System.out.print( s2.peek() );
```

A. `12` B. `5` C. `13`

D. `17` E. `-5`

QUESTION 29

What is output by the following client code?

```
Structure s3 = new Structure();
int[] els = {12, 5, 13, 17, -5};
for(int i : els )
  s3.add( i );
s3.show();
```

A. `-55121317`

B. `17135-512`

C. `5-5131217`

D. `1713125-5`

E. `513-51712`

QUESTION 30

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

A. A stack.

B. A list.

C. A max heap.

D. A hash table.

E. A binary search tree.

```
public class Node{
  public Node pt, rt, lt;
  public int dt;
  public Node(Node p, int d){ pt = p; dt = d; }
}

public class Structure{
  private int size;
  private Node start;

  public void add(int d){
    String path = findP(++size).substring(1);
    if( size == 1 )
      start = new Node(null, d);
    else
      addHelp(path, d);
  }

  private void addHelp(String p, int d){
    Node tp = start;
    while( p.length() > 1 ){
      tp = p.charAt(0) == '0' ? tp.lt : tp.rt;
      p = p.substring(1);
    }
    if( p.equals("0")){
      tp.lt = new Node(tp, d); tp = tp.lt;
    } else {
      tp.rt = new Node(tp, d); tp = tp.rt;
    }
    adjust(tp);
  }

  private void adjust(Node tp){
    int temp;
    while(tp.pt != null && tp.dt > tp.pt.dt){
      temp = tp.dt;
      tp.dt = tp.pt.dt;
      tp.pt.dt = temp;
      tp = tp.pt;
    }
  }

  public String findP(int t){
    if(t == 0 || t == 1)
      return t + "";
    else
      return findP( t / 2 ) + (t % 2);
  }

  private void show(Node n){
    if(n != null){
      System.out.print(n.dt);
      show(n.lt);
      show(n.rt);
    }
  }
  public void show(){  show(start); }
  public int peek(){ return start.dt; }
}
```

QUESTION 31

Which of the following replaces **<*1>** in the code to the right to end method `sort` if the Boolean expression is `true`?

A. `break`

B. `continue`

C. `exit`

D. `end`

E. `return`

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

QUESTION 32

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

A. `403-1786`

B. `-1034678`

C. `-1034786`

D. `467830-1`

E. `6034-187`

QUESTION 33

What sorting algorithm is implemented by method `sort`?

A. Mergesort

B. Heapsort

C. Quicksort

D. Selection sort

E. Insertion sort

QUESTION 34

Given an array of N unique integers in random order, what is the running time of method `sort`? Assume the `print` method is O(1). Choose the most restrictive correct answer.

A. O(N)

B. O(NlogN)

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

D. O(N^2)

E. O(N!)

```
public static void swap( int[] a,
                          int i, int j) {
  int tmp = a[i];
  a[i] = a[j];
  a[j] = tmp;
}


public static void sort( int[] list,
                   int start, int stop ){
  if(start >= stop)
    <*1>;

  int p = (start + stop) / 2;

  swap(list, p, start);
  int pVal = list[start];

  int i, j = start;

  for(i = start + 1; i <= stop; i++ ){
    if( list[i] <= pVal){
      j++;
      swap(list, i, j);
    }
  }

  if(start == 0 && stop == list.length - 1)
    for(int v : list)
      System.out.print(v);

  swap(list, start, j);
  sort( list, start, j - 1 );
  sort( list, j + 1, stop );
}


public static void sample(){
  int[] data = {6, 0, 3, 4, 7, 8, -1};
  sort(data, 0, 6);
}
```

QUESTION 35

Which of the following best describes what method `eval` returns?

A. The minimum of the three parameters.

B. The maximum of the three parameters.

C. How many of the three parameters equal each other.

D. The range of the three parameters.

E. The sum of the three parameters.

QUESTION 36

Which of the following replaces **`<*1>`** in the code to the right to create a two dimensional array of `ints` with one more row than the number of characters in the `String s` and one more column than then number of characters in the `String t`?

A. `int[s.length()][t.length()]`

B. `new int[t.size()+1][s.size()+1]`

C. `new int[i + 1][j + 1]`

D. `new int[n + 1][m + 1]`

E. More than one of these.

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

QUESTION 37

What is returned by the method call `comp("uilcs", "uilcs")` ?

A. 6

B. 5

C. 3

D. 2

E. 0

QUESTION 38

What is returned by the method call `comp("state", "stilte")` ?

A. 2

B. 3

C. 4

D. 5

E. 6

```
private static int eval (int a, int b, int c){
  int m;
  m = a;
  if (b < m)
    m = b;
  if (c < m)
    m = c;
  return m;
}

public static int comp (String s, String t){
  int d[][];
  int n, m, i, j;
  char s_i, t_j, cost;

  n = s.length ();
  m = t.length ();
  if (n == 0)
    return m;
  if (m == 0)
    return n;

  d = <*1>;
  for (i = 0; i <= n; i++)
    d[i][0] = i;

  for (j = 0; j <= m; j++)
    d[0][j] = j;

  for (i = 1; i <= n; i++) {
    s_i = s.charAt (i - 1);
    for (j = 1; j <= m; j++) {
      t_j = t.charAt (j - 1);

      if (s_i == t_j)
        cost = 0;
      else
        cost = 1;

      d[i][j] = eval(d[i-1][j]+1,
             d[i][j-1]+1, d[i-1][j-1] + cost);
    }
  }
  return d[n][m];
}
```

QUESTION 39

When implementing a hash table what are the two most common collision resolution schemes?

A. heapify and post order

B. open addressing and chaining

C. serialization and cloning

D. constructors and interfaces

E. keying and perfect hashing

QUESTION 40

You are working with an existing sort method that sorts data into ascending order, but you do not have the source code for the method. Here are some results of experiments with the sorting method:

Time to sort an array of 10,000 elements in random order: 0.5 seconds
Time to sort an array of 20,000 elements in random order: 2.0 seconds

Time to sort an array of 10,000 elements already in ascending order: 0.5 seconds
Time to sort an array of 20,000 elements already in ascending order: 2.0 seconds

Based on these results, which sorting algorithm is most likely being used?

A. Insertion sort

B. Selection sort

C. Mergesort

D. Quicksort

E. Radixsort

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`
- `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 java.lang.Comparable`

- `Integer(int value)`
  // constructor
- `int intValue()`
- `boolean equals(Object other)`
- `String toString()`
- `int compareTo(Integer anotherInteger)`
  // specified by `java.lang.Comparable`
- `static int parseInt(String s)`
  // Parses the string argument as a signed decimal integer.

`class java.lang.Double implements java.lang.Comparable`
- `Double(double value)`
  // constructor
- `double doubleValue()`
- `boolean equals(Object other)`
- `String toString()`
- `int compareTo(Double anotherDouble)`
  // specified by `java.lang.Comparable`

`class java.lang.String implements java.lang.Comparable`
- `int compareTo(String anotherString)`
  // specified by `java.lang.Comparable`
- `boolean equals(Object other)`
- `int length()`
- `String substring(int from, int to)`
  // returns the substring beginning at `from`
  // and ending at `to-1`
- `String substring(int from)`
  //returns `substring(from, length())`
- `int indexOf(String s)`
  // returns the index of the first occurrence of `s;`
  // returns `-1` if not found
- `int indexOf(String str, int fromindex)`
  // Returns the index within this string of the first occurrence
  // of the specified substring, starting the
  // search at the specified index.`char`
- `charAt(int index)`
  // Returns the character at the specified index.
- `int indexOf(int ch)`
  // Returns the index within this string of the first occurrence
  // of the specified character.
- `int indexOf(int ch, int fromindex)`
  // Returns the index within this string of the first occurrence
  // of the specified character, starting the
  // search at the specified index.
- `String toLowerCase()`
  // Converts all of the characters in this `String` to lower
  // case using the rules of the default locale.
- `String toUpperCase()`
  // Converts all of the characters in this `String` to upper
  // case using the rules of the default locale.
- `String[] split(String regex)`
  // Splits this string around matches of the given regular
  // expression.

`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 x)`
- `static double abs(double x)`
- `static double pow(double base, double exponent)`
- `static double sqrt(double x)`
- `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)`

`class java.util.Random`
- `int nextInt()`
- `double nextDouble()`

`interface java.util.List<E>`
- `boolean add(E x)`
- `int size()`
- `Iterator<E> iterator()`
- ListIterator<E> listIterator()

class java.util.ArrayList<E> implements java.util.List<E>

- Methods in addition to the `List` methods:
- `E get(int index)`
- `E set(int index, E x)`
  // replaces the element at `index` with `x`
- `void add(int index, E x)`
  // inserts `x` 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 java.util.List<E>

- Methods in addition to the `List` methods
- `void addFirst(E x)`
- `void addLast(E x)`
- `E getFirst()`
- `E getLast()`
- `E removeFirst()`
- `E removeLast()`

interface java.util.Set<E>

- `boolean add(E x)`
- `boolean contains(Object x)`
- `boolean remove(Object x)`
- `int size()`
- `Iterator<E> iterator()`

class java.util.HashSet<E> implements java.util.Set<E>

class java.util.TreeSet<E> implements java.util.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 java.util.Map<K,V>

class java.util.TreeMap<K,V> implements java.util.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 x)`
- `void set(E x)`

class java.lang.StringBuffer

- `StringBuffer append(char c)`
- `StringBuffer append(String str)`
- `StringBuffer append(StringBuffer sb)`
- `int capacity()`
- `char charAt(int index)`
- `StringBuffer delete(int start, int end)`
- `StringBuffer deleteCharAt(int index)`
- `StringBuffer insert(int offset, char c)`
- `StringBuffer insert(int offset, String S)`
- `int length()`
- `void setCharAt(int index, char ch)`
- `String substring(int start)`
- `String substring(int start, int end)`
- `String toString()`

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)`
  `//` Sets this scanner's delimiting pattern to a pattern
  `// constructed from the specified`
  `// String.`

Computer Science Answer Key UIL State 2007

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

12. E. Choices A and B are both correct. Since this is the Score class we have access to all Score objects private fields.

14. C. The ints returned from `Object`'s `hashCode` may be different from one run of an application to the next.

16. B. The sum of the geometric sequence n + n/2 + n/4 + ... + 4 + 2 + 1 = 2n - 1.

19. D. ~ is the bitwise invert operator. 2 -> 0000 0010 : ~2 -> 1111 1101 = -3 in 2s complement.
(128 - 125 = 3, sign bit indicates negative.)

23. C. Method find returns the index of the first occurrence of an element in the array using a binary search to find an occurrence and then a linear search to find the first occurrence. If all elements are equal to the same value the linear search traverses half of the array.

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.