University Interscholastic League

Computer Science Competition

Number 112 (State - 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 745_8 and 1101101_2?

A. 900_{16} B. 11111111_2 C. 252_{16} D. 900_{10} E. $B22_{16}$

QUESTION 2

What is output by the code to the right?

A. 30 B. 15 C. 12

D. 312 E. 18

```
int s = 3;
int t = s + 2 * s;
System.out.println( s + t );
```

QUESTION 3

What is output by the code to the right?

A. 10 B. 9 C. 22

D. 18 E. 20

```
int accum = 0;
for(int i = 1; i <= 10; i++){
  accum++;
  accum++;
}
System.out.println( accum );
```

QUESTION 4

What is output by the code to the right?

A. 4 B. 10 C. 9

D. 3 E. 5

```
String first = "Doug";
String second = "Burger";
first += second;
System.out.println( first.length() );
```

QUESTION 5

What is output by the code to the right?

A. 13 B. 15 C. 17

D. 16 E. 10

```
int[] ps = {2, 3, 5, 7, 11};
ps[1] += ps[0] + 3 + ps.length;
System.out.println( ps[1] );
```

QUESTION 6

What is output by the code to the right?

A. 14 B. 6 C. 2

D. 6.0 E. 2.33333333

```
int x = 14;
int y = 2;
x /= 3 * y;
System.out.println( x );
```

QUESTION 7

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

A. 7 B. 8 C. 2

D. 4 E. 1

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

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

QUESTION 8

What is output by the code to the right?

A. `12` B. `13` C. `1`

D. `2` E. `3`

```
String ans = "ABBAACDC";
if( ans.equals( "ABBA" ) )
  System.out.println( 1 );
if( ans.length() > 4 )
  System.out.println( 2 );
else
  System.out.println( 3 );
```

QUESTION 9

Which class is `Rectangle`'s super class?

A. `Shape`

B. `Object`

C. `Square`

D. `String`

E. `Rectangle` does not have a super class.

QUESTION 10

What is output by the client code to the right?

A. `r1`

B. `this`

C. `6`

D. `0`

E. The output cannot be determined until the program is run.

```
public class Rectangle{

  private int width;
  private int height;

  public Rectangle(int w, int h){
    width = w;
    height = h;
  }

  public int area(){
    return width * height;
  }

  public String toString(){
    return "" + this.area();
  }
}

// client code
Rectangle r1 = new Rectangle( 2, 3 );
System.out.println( r1 );
```

QUESTION 11

What is output by the code to the right?

A. `13` B. `63` C. `21`

D. `-31` E. `26`

```
int m = 63;
int n = 42;
int o = n ^ m;
System.out.print( o );
```

QUESTION 12

What is output by the code to the right?

A. `4.0` B. `1` C. `4.4`

D. `5.0` E. `5`

```
double p = 1.1;
System.out.print( Math.ceil( p * 4 ) );
```

QUESTION 13

What is output by the code to the right?

A.
```
linelineline
```
B.
```
lineline
line
```
C.
```
line
lineline
```
D.
```
line/nlineline
```
E.
```
line
line
line
```

```
System.out.print( "line\n" );
System.out.println( "line" );
System.out.print( "line" );
```

QUESTION 14

What is output by the code to the right?

A. `(012)` B. `00010` C. `00012`

D. `(((12` E. `(00012)`

```
String format = "%0(5d";
int val = 12;
System.out.printf( format, val );
```

QUESTION 15

What is returned by the method call `change( change(8) )` ?

A. `10` B. `6` C. `13`

D. `4` E. `8`

```
public static int change(int x){
  int y = x;
  y = y - 2;
  x /= 2;
  return x + y;
}
```

QUESTION 16

What is output by the code to the right?

A. `_gr` B. `_gray` C. `m_`

D. `_gra` E. `_g`

```
String name = "jim_gray";
String st;
st = name.substring(1, 5).substring(2);
System.out.print( st );
```

QUESTION 17

What is output by the code to the right?

A. `:` B. `71` C. `7`

D. `14` E. `*&^`

```
String messy;
messy = "6...56fg@71*&^14:";
String[] res = messy.split( "\\D+" );
System.out.print( res[3] );
```

QUESTION 18

What is output by the code to the right?

A. `7`

B. `13`

C. `6`

D. `5`

E. `9`

```
String data = "MAURICEWILKES";
int count = 0;
for(int i = 0; i < data.length(); i++){
  if( data.charAt( i ) < 'J' )
    continue;
  count++;
  i++;
}
System.out.print( count );
```

QUESTION 19

What is output by method `over` if `input` initially contains the following `Strings`?

`["Perl", "Cobol", "Fortran", "Ruby"]`

A. `Cobol`

B. `Fortran`

C. There is no output due to a syntax error in method `over`.

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

E. There is no output due to an infinite loop caused by a logic error in method `over`.

```
public static void over
                 (ArrayList<String> input){

  Iterator<String> it = input.iterator();
  boolean found = false;
  while( it.hasNext() ){
    found = found
        || it.next().length() > 4;
  }
  System.out.println( it.next() );
}
```

QUESTION 20

What replaces **<*1>** in the code to the right to indicate `numRight` is not defined in the `TestScore` class and that subclasses are responsible for defining it?

A. `static`

B. `interface`

C. `abstract`

D. `abstract extends`

E. `extends`

```
public abstract class TestScore{

  public <*1> int numRight();

  public int numQuestions(){
    return 100;
  }

  public int score(){
    double raw = numRight();
    raw = raw / numQuestions() * 100;
    return (int)raw;
  }
}

public class ACTScore extends TestScore{
  private int correct;
  private int questions;

  public ACTScore(int c, int q){
    correct = c;
    questions = q;
  }

  public int numRight(){
    return correct;
  }

  public int numQuestions(){
    return questions;
  }

  public void adjustScore(int adj){
    correct += adj;
  }

}
```

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

QUESTION 21

What is output by the following client code?

```
ACTScore sc = new ACTScore( 50, 100 );
System.out.print( sc.score() );
```

A. `0.5` B. `100` C. `50`

D. `1.0` E. `0.0`

QUESTION 22

What is output by the following client code?

```
ACTScore as = new ACTScore( 145, 200 );
TestScore ts = as;
as.adjustScore( 5 );
System.out.print( ts.score() );
```

A. `75`

B. `150`

C. `145`

D. 72

E. There is no output due to a syntax error in the client code.

QUESTION 23

What is output by the code to the right?

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

D. `5` E. `3`

```
int j = 5;
int k = (j++ > 5) ? 4 : ((j > 4) ? 3 : 2);
System.out.println( k );
```

QUESTION 24

What is output by the code to the right?

A. `[AAA, AB]` B. `[AA, A, B, A]`

C. `[AA, A, B]` D. `[AA, B]`

E. `[AA, B, A]`

```
ArrayList<String> letters = new
                       ArrayList<String>();
letters.add( "A" );
letters.add( "A" );
letters.add( 1, "B" );
letters.set( 0, "AA" );
System.out.println( letters );
```

QUESTION 25

What is output by the statement marked `line 1` when method `demo` is called?

A. `0` B. `5`

C. `3` D. `1`

E. `12208`

QUESTION 26

What is output by the statement marked `line 2` when method `demo` is called?

A. `24` B. `9`

C. `5` D. `7`

E. `15`

```java
public static int sum(int[][] mat,
                                  int row, int col){
  if( row == -1 || col == mat[0].length )
    return 0;

  int tot1 = sum(mat, row - 1, col);
  int tot2 = sum(mat, row - 1, col + 1);
  return tot1 + tot2 + mat[row][col];
}

public static void demo(){
  int[][] t = { {1, 2, 2, 0, 8},
                {5, 1, 0, 1, 2},
                {4, 2, 3, 2, 1}};

  System.out.println( t.length ); // line 1

  int x = sum( t, 2, 1 );

  System.out.println( x ); // line 2
}
```

QUESTION 27

Which searching algorithm does method `search` implement?

A. Greedy B. Sequential C. Heap

D. Binary E. Interpolation

QUESTION 28

Which of the following best describes method `search`'s post condition ?

A. Returns the index of the first occurrence of `tgt` in `nums`.

B. Returns the index of the first occurrence of `tgt` in `nums` or `-1` if `tgt` is not present.

C. Returns the number of times `tgt` appears in `nums`.

D. Returns the number of elements in `nums` that are less than `tgt`.

E. Returns the index of the last occurrence of `tgt` in `nums` or `-1` if `tgt` is not present.

```java
// pre: nums != null
// post: see question 28
public static int search(int[] nums,
                                        int tgt){
  int result = -1;
  for(int i = 0; i < nums.length; i++){
    if( nums[i] == tgt ){
      result = i;
    }
  }
  return result;
}
```

QUESTION 29

What is output by the code to the right?

A. `aan_emer` B. `lb_emer` C. `n_emer`

D. `bn_emer` E. `alan_emer`

```java
/* explanation of method replaceFirst:
   String replaceFirst(String regex,
                            String replacement)
   Creates and returns a new String by
   replacing the first substring that
   matches regex with replacement.
*/

String nm = "alan_emer";
String regex = "a..";
String rs = nm.replaceFirst( regex, "b" );
System.out.print( rs );
```

QUESTION 30

Consider method `build` to the right. When the parameter `n` equals 500,000 the method takes 1.5 seconds to complete. What is the expected time for method `build` to complete when `n` equals 1,000,000?

A. 6.0 sec. B. 2.25 sec. C. 3.15 sec.

D. 1.5 sec. E. 3.0 sec.

```
public TreeSet<Integer> build(int n){
  TreeSet<Integer> tree
                    = new TreeSet<Integer>();
  for(int i = 0; i < n; i++)
    tree.add( i );
  return tree;
}
```

QUESTION 31

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

A. `6`

B. `9`

C. `3`

D. There is no output due to a syntax error in method `alter`.

E. There is not output due to an `ArrayIndexOutOfBoundsException`.

```
public static int alter(int[] data){
  int i = 0;
  try{
    while( i < data.length )
      data[i++] *= 2;
    if( i > 2 )
      return i;
    return 0;
  }
  finally{
    data[1] += 3;
  }
}

public static void initiate(){
  int[] data = {5, 3, 5, 2, 1, 7, 8};
  alter( data );
  System.out.println( data[1] );
}
```

QUESTION 32

If N equals `values.length` what is the Big O of method `fill` when `res` is an `ArrayList` and when `res` is a `LinkedList`? Pick the most restrictive correct set of answers.

	`ArrayList`	`LinkedList`
A.	O(N)	O(N)
B.	O(N^3)	O(N^2)
C.	O(N^2)	O(N^2)
D.	O(N^2)	O(N)
E.	O(N)	O(N^2)

```
// pre: res.size() == 0
public static void fill(List<Integer> res,
                              int[] values){

    for(int element : values)
      res.add( 0, element );
}
```

QUESTION 33

What is returned by the method call `beta(20)`?

A. `-4` B. `-10` C. `0`

D. `-9` E. `-7`

```
public static int alpha(int x){
  return (x < 0) ? x * 2 : beta( x + 10 );
}

public static int beta(int y){
  return alpha( y - 15 ) + 1;
}
```

QUESTION 34

What replaces `<*1>` in the code to the right so that the body of the `if` statement executes when the element at position `i` in the array `d` is less that or equal to the variable `guide` according to the natural ordering of the elements of `d`?

A. `d[i].compareTo( guide ) >= 0`

B. `guide.compareTo( d[i] ) <= 0`

C. `!guide.equals( d[i] )`

D. `d[i] <= guide`

E. `d[i].compareTo( guide ) <= 0`

```
public static void sort(Comparable[] d){
  int start, end;
  Stack<Integer> sp = new Stack<Integer>();
  sp.push( 0 );
  sp.push( d.length - 1 );
  while( !sp.isEmpty() ){
    end = sp.pop();
    start = sp.pop();
    if(start < end){
      int p = ( start + end ) / 2;
      swap( d, p, start );
      Comparable guide = d[start];
      int i, j = start;
      for(i = start + 1; i <= end; i++ ){
        if( <*1> ){
          j++;
          swap( d, i, j );
        }
      }
      swap( d, start, j );
      sp.push( start );
      sp.push( j - 1 );
      sp.push( j + 1 );
      sp.push( end );
    }
  }
}

public static void swap(Comparable[] d,
                             int i, int j){
  Comparable t = d[i];
  d[i] = d[j];
  d[j] = t;
}
```

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

QUESTION 35

Which sorting algorithm do methods `sort` and `swap` implement?

A. Merge sort

B. Insertion sort

C. Quicksort

D. Selection sort

E. Stack sort

QUESTION 36

The depth of a node in a tree is defined as the number of links from the root node of the tree to that node. The depth of the root node is 0.

The following values are inserted one at a time in the order shown, from left to right, into a binary search tree using the traditional insertion algorithm.

`25   1   13   -5   100   12   50   10   -7   200   8`

What is the depth of the node that contains `12` in the resulting tree?

A. 0   B. 11   C. 3   D. 5   E. 2

QUESTION 37

What is output by the code to the right?

A. -1   B. 255   C. 0

D. 128   E. -16777216

```
int num = -1;
System.out.println( num >>> 24 );
```

QUESTION 38

What is output by the following client code?

```
int result = Structure.value( "cab" );
System.out.print( result );
```

A. 2

B. 3

C. 6

D. 0

E. 294

QUESTION 39

What is output by the following client code?

```
Structure ds = new Structure();
ds.add( "cab" );
ds.add( "aaa" );
ds.add( "dead_e" );
ds.add( "bac" );
ds.add( "ACM" );

ds.showAll();
```

A. aaa_cab_bac_dead_e_ACM_

B. ACM_aaa_bac_cab_dead_e_

C. ACM_aaa_bac_cab_deade_

D. ACM_aaa_cab_bac_dead_e_

E. cab_aaa_ACM_bac_dead_e_

QUESTION 40

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

A. An array based list

B. A stack

C. A set

D. A hash table

E. A min heap

```
public class Structure{

  private static final int LIM = 125;

  private Object[] con;

  public Structure(){
    con = new Object[LIM];
    for(int i = 0; i < LIM; i++)
      con[i] = new ArrayList<String>();
  }

  public void add(String str){
    int spot = value( str );
    get( spot % LIM ).add( str );
  }

  public void showAll(){
    for( Object temp : con ){
      ArrayList<String> list =
                      (ArrayList<String>)temp;
      for( String str : list )
        System.out.print( str + "_" );
    }
  }

  public boolean contains(String str){
    int spot = value( str ) % LIM;
    return get( spot ).contains( str );
  }

  public boolean remove(String str){
    int spot = value( str ) % LIM;
    return get( spot ).remove( str );
  }

  private ArrayList<String> get(int p){
    return (ArrayList<String>)con[p];
  }

  public static int value(String str){
    int t = 0;
    char c;
    for(int i = 0; i < str.length(); i++){
      c = str.charAt( i );
      if( Character.isLetter(c) )
        t += Character.toLowerCase(c) - 'a';
    }
    return t;
  }
}
```

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 State 2008

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

14. The `0` flag causes the result to be padded with `0`s. The `(` flag causes <u>negative</u> numbers to be enclosed in parenthesis. It does not affect positive numbers.

26. The algorithm causes the 2 at row 0, column 2 to be counted twice.

30. `TreeSet` uses a balanced binary search tree. Adding N elements in order is still O(NlogN). Thus when the number of elements is doubled, the time should increase by a little more than a factor of two.

31. The `finally` block is executed before the `return` in method `alter` is carried out

37. Negative numbers are stored in 2's complement format. `-1` results in all bits being set to 1. `>>>` is the logical right shift operator which causes the sign bit to be shifted. At the bit level -1 is `11111111 11111111 11111111 1111111.` Shifting in 24 0's results in `00000000 00000000 00000000 11111111` which is the representation of `255`.

40. The data structure is not strictly a set because the data structure may contain multiple copies of the same `String`.