University Interscholastic League

Computer Science Competition

Number 124 (State - 2010)

General Directions (Please read carefully!):

1) DO NOT OPEN EXAM UNTIL TOLD TO DO SO.

2) **NO CALCULATOR OF ANY KIND MAY BE USED.**

3) There are 40 questions on this contest exam. You have 45 minutes to complete this contest. If you are in the process of actually writing an answer when the signal to stop is given, you may finish writing that answer.

4) Papers may not be turned in until 45 minutes have elapsed. If you finish the test before the end of the allotted time, remain at your seat and retain your paper until told to do otherwise. Use this time to check your answers.

5) All answers must be written on the answer sheet/Scantron card provided. Indicate your answers in the appropriate blanks provided on the answer sheet or on the Scantron card. Clean erasures are necessary for accurate Scantron grading.

6) You may place as many notations as you desire anywhere on the test paper, but not on the answer sheet or Scantron card which are reserved for answers only.

7) You may use additional scratch paper provided by the contest director.

8) All questions have ONE and only ONE correct (BEST) answer. There is a penalty for all incorrect answers. **All provided code segments are intended to be syntactically correct, unless otherwise stated. Ignore any typographical errors and assume any undefined variables are defined as used.**

9) A reference to commonly used Java classes is provided at the end of the test, and you may use this reference sheet during the contest. You may detach the reference sheets from the test booklet, but DO NOT DO SO UNTIL THE CONTEST BEGINS.

10) Assume that any necessary import statements for standard Java packages and classes (e.g. `.util`, `ArrayList`, etc.) are included in any programs or code segments that refer to methods from these classes and packages.

Scoring:

1) All questions will receive **6 points** if answered correctly; no points will be given or subtracted if unanswered; **2 points** will be deducted for an incorrect answer.

QUESTION 1

What is the sum of 64_8 and 55_8?

A. 111_2 B. 11001_2 C. 1100001_2 D. 111001_2 E. 1100111_2

QUESTION 2

What is output by the code to the right?

A. 5.0 B. 5.25 C. 5.5

D. 0.75 E. 9.0

```
int x = 10;
double a = 2.5;
a = a / x + a * 2;
System.out.print(a);
```

QUESTION 3

What is output by the code to the right?

A. 7 5 B. 5 8 C. 8 5

D. 7 7 E. 5 5

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

QUESTION 4

What is output by the code to the right?

A. 4 B. 8 C. 16

D. 20 E. 24

```
String lang = "Naur";
String big = lang + lang;
big = big + big + lang;
System.out.print(lang.length());
```

QUESTION 5

What is output by the code to the right?

A. 12403 B. 12106 C. 42103

D. 45403 E. 15103

```
int[] small = {1, 2, 1, 0, 3};
small[small[small[0]]] += 3;
for(int i : small)
  System.out.print(i);
```

QUESTION 6

What is output by the code to the right?

A. 12 B. 10 C. 8

D. 6 E. 4

```
int w = 4;
int z = 2;
z = 2 + 2 * ++w;
System.out.print(z);
```

QUESTION 7

Which answer is logically equivalent to the following `boolean` expression, where `p`, `q`, and `r` are `boolean` variables?

```
!(!p || q) || (r && !r)
```

A. `(p || !q) || (r && !r)` B. `r` C. `!r`

D `(p || q)` E. `p && !q`

QUESTION 8

What is output by the code to the right?

A. 13 B. 12 C. 2

D. 3 E. 1

```
int x = 10;
int y = 20;
if( x % y > y % x )
  System.out.print(1);
if( x / y > y / x )
  System.out.print(2);
else if( y > x )
  System.out.print(3);
```

QUESTION 9

What replaces **<*1>** in the code to the right to pass the calling object as an argument?

A. distance B. HomeRun C. super

D. other E. this

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

```
public class HomeRun{

  private int distance;

  public HomeRun(int d){
    distance = d;
  }

  public boolean isLonger(HomeRun other){
    return distance > other.distance;
  }

  public boolean same(HomeRun other){
    return !isLonger(other) &&
      !other.isLonger(<*1>);
  }
}
```

QUESTION 10

What is output by the following client code?

```
HomeRun h1 = new HomeRun(500);
HomeRun h2 = new HomeRun(50 * 10);
System.out.print( h1.same(h2) );
```

A. false B. true C. 0

D. 1 E. equals

QUESTION 11

What is output by the code to the right?

A. 30 3 B. 30 240 C. 3 3

D. 240 240 E. 3 30

```
int ax = 30;
int bx = ax << 3;
System.out.print(ax + " " + bx);
```

QUESTION 12

What is output by the code to the right?

A. 5 12 B. -5 12 C. 5 10

D. -5 5 E. -5 10

```
int st = Math.min(-5, 5);
int res = Math.max(Math.min(12, 10), st);
System.out.print(st + " " + res);
```

QUESTION 13

What is output by the code to the right?

A. Red Blue Green

B. Red Blue Green

C. Red
Blue
Green

D. Red
Blue Green

E. RedBlueGreen

```
System.out.print("Red\t");
System.out.print("Blue\t");
System.out.println();
System.out.print("Green");
```

QUESTION 14

What is output by the code to the right?

A. `3.1 3.142` B. `3.13.1`

C. `3.1 3.1` D. `3.13.14`

E. `3.1416 3.14`

```
double val = 3.14159;
String format = "%1$3.1f %2$3.3f";
System.out.printf(format, val, val);
```

QUESTION 15

What is returned by the method call `hat(hat(2, 3), hat(1, 1))`?

A. `-2` B. `1` C. `2`

D. `3` E. `4`

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

QUESTION 16

What is output by the code to the right?

A. `false false` B. `false true`

C. `true true` D. `1 1`

E. `true false`

```
boolean p;
boolean q;
String nm1 = "Shannon";
String nm2 = "Shortliffe";
String nm3 = "Rivest";
p = nm1.compareTo(nm2) < 0;
q = nm2.compareTo(nm3) > 0;
System.out.print( p + " " + q);
```

QUESTION 17

What is output by the code to the right?

A. `400` B. `210` C. `120`

D. `40` E. `20`

```
String result = "";
for(int i = 0; i < 20; i++)
  for(int j = i; j < 20; j++)
    result = result + "*";
System.out.print( result.length() );
```

QUESTION 18

The code to the right contains a syntax error. Which line of code causes the syntax error?

A. `line 1`

B. `line 2`

C. `line 3`

D. `line 4`

E. `line 5`

```
ArrayList<String> data; // line 1
data = new ArrayList<String>(); // line 2
data.add(0, "12"); // line 3
data.add(12); //line 4
data.add(12 + ""); // line 5
```

QUESTION 19

What is output by the code to the right?

A. `[5, 3]`

B. `[5, 0, 3]`

C. `[0, 5, 3]`

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

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

```
List<Integer> list1;
list1 = new ArrayList<Integer>();
list1.add(5);
list1.add(3);
List<Integer> list2 = list1;
list2.add(1, 0);
System.out.print(list1);
```

QUESTION 20

What is output by the code to the right?

A. 0 B. 6 C. 12

D. 36 E. 64

```
int val = 1;
for(int i = 0; i < 6; i++)
  val *= 2;
System.out.print(val);
```

QUESTION 21

What is output by the code to the right?

A. 15.0 B. 125.0 C. 243.0

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

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

```
System.out.print(Math.pow(5, 3));
```

QUESTION 22

What is output by the code to the right?

A. 7 2 B. 3 1 C. 3 3

D. 3 2 E. 0 1

```
int x = 3;
int y = 7 / 2;
int z = (x > y) ? (x < y) ? 1 : 2 : 3;
System.out.print(y + " " + z);
```

QUESTION 23

Given the `Readable` interface and the `EBook` class, what is output by the following client code?

```
EBook e1 = new EBook(2000);
System.out.print( e1 );
```

A. 2000 B. word g

C. 2000 null D. 2000 EBook

E. 2000 genre

QUESTION 24

Given the `Readable` interface and the `EBook` class, what is output by the following client code?

```
Readable r1 = new EBook(5000);
System.out.print(r1.hashCode());
```

A. -1 B. 0 C. 1

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

E. The output will vary from one run of the program to the next.

QUESTION 25

Given the `Readable` interface and the `EBook` class, what is output by the client code to the right?

A. true null B. true true

C. true false D. false true

E. false false

```
public interface Readable{
  public int numWords();
  public String genre();
}

public class EBook implements Readable{
  private static final String g = "EBook";
  private int words;

  public EBook(int w){
    words = w;
  }

  public String genre(){ return g; }

  public int numWords(){ return words; }

  public boolean equals(EBook obj){
    return this.words == obj.words;
  }

  public String toString(){
    return words + " " + g;
  }
}

// client code Question 25
Object obj1 = new EBook(1000);
Object obj2 = new EBook(1000);
System.out.print( obj1 == obj2 );
System.out.print( " " + obj1.equals(obj2));
```

QUESTION 26

Which of the following statements regarding `abstract` classes, such as `Mammal` to the right, is false?

A. Declarations such as

`Mammal m;`

do not cause syntax errors.

B. The class may have constructors.

C. The class may have instance variables.

D. The class may have private methods.

E. The class must have at least one method that is declared `abstract`.

```
public abstract class Mammal {
  // implementation details of class
  // not shown.
}
```

QUESTION 27

Which of the following can replace **<*1>** in the code to the right so that method `sort(int[], int)` correctly sorts the elements of `data` into ascending order?

I. `sort(data, i + 1)`
II. `sort(data, i ^ 2)`
III. `sort(data, i >> 1)`

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

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

```
public void sort(int[] data){
  sort(data, 0);
}

public void sort(int[] data, int i){
  if(i < data.length - 1){
    int j = getMinIndex(data, i);
    int temp = data[j];
    data[j] = data[i];
    data[i] = temp;
    <*1>;
  }
}

public int getMinIndex(int[] data, int i){
  if(i == data.length - 1)
    return i;
  int j = getMinIndex(data, i + 1);
  if( data[i] < data[j] )
   return i;
  return j;
}
```

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

QUESTION 28

Which sorting algorithm do the methods `sort(int[])`, `sort(int[], int)`, and `getMinIndex(int[], int)`, implement?

A. insertion sort

B. selection sort

C. heap sort

D. quicksort

E. merge sort

QUESTION 29

What is output by the code to the right?

A. `1234` B. `32512` C. `0123`

D. `12345` E. `vvvv`

```
int[] vals = {3, 2, 5, 12};
for(int v : vals)
  System.out.print(v);
```

QUESTION 30

What is output by the code to the right?

A. `00`

B. `010`

C. `100`

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

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

```
ArrayList<List<String>> names;
names = new ArrayList<List<String>>();
names.add(new ArrayList<String>());
names.add(new LinkedList<String>());
for(List<String> ns : names)
  System.out.print(ns.size());
```

QUESTION 31

Consider method `makeBigNum` to the right. When `n is` equal to `10000` it takes 2 seconds to complete. What is the expected time for the method to complete when `n` is equal to `20000`?

A. 0.5 seconds B. 8 seconds C. 6 seconds

D. 2 seconds E. 4 seconds

```
public String makeBigNum(int n){
  String result = "1";
  for(int i = 0; i < n; i++)
    result += "0";
  return result;
}
```

QUESTION 32

What replaces **<*1>** in the code to the right to add the value stored in `t2` to the `ArrayList b`, at position `p`?

A. `b.add(p, t2)` B. `add(b, p, t2)`

C. `b.add(t2)` D. `b.what(p, t2)`

E. `Collection.add(b, p, t2)`

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

QUESTION 33

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

A. `1` B. `3`

C. `4` D. `6`

E. `7`

QUESTION 34

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

A. `8` B. `6`

C. `3` D. `1`

E. `0`

```
public int what(ArrayList<Integer> a,
                ArrayList<Integer> b) {
  int r = 0;
  if(a.size() == 0 || b.size() == 0)
    r = Math.abs(a.size() - b.size());
  else {
    int t1 = a.remove(a.size() - 1);
    int p = b.indexOf(t1);
    int t2 = 0;
    if( p >= 0 )
      t2 = b.remove(p);
    else
      r = 1;
    r += what(a, b);
    a.add(t1);
    if(p >= 0)
      <*1>;
  }
  return r;
}

// client code
int[] list1 = {3, 5, 1, 3, 5, 6};
int[] list2 = {5, 3, 9, 5, 1, 4, 1, 3};
ArrayList<Integer> a, b;
a = new ArrayList<Integer>();
b = new ArrayList<Integer>();

for(int i : list1)
  a.add(i);
for(int i : list2)
  b.add(i);

System.out.print( what(a, b) ); // line 1

System.out.print( a.size() ); // line 2
```

QUESTION 35

What is the Big O of method `process` to the right? Assume N equals `data.length`. Pick the most restrictive correct answer.

A. O(logN) B. O(N)

C. O(NlogN) D. $O(N^2)$

E. $O(N^2 logN)$

```
public LinkedList<String> process(
                              String[] data) {
  LinkedList<String> result;
  result = new LinkedList<String>();

  for(int i = 0; i < data.length; i++)
    result.addFirst( data[i] );

  return result;
}
```

QUESTION 36

The following values are added one at a time in the order shown to a binary search tree using the traditional algorithm. What is the height of the resulting tree? The height of a tree is the number of links from the root node to the deepest leaf in the tree. A tree with a single node, the root node, has a height of 0.

`9, 3, 9, 0, 5, 14, -5, -7, 12, 5, 0`

A. `8` B. `7` C. `6` D. `5` E. `4`

QUESTION 37

What is returned by the method call `count(list)` if `list` contains the values shown below?

`{"AA", "B", null, "CA", null, "CCC"}`

A. `0`

B. `-1`

C. `3`

D. `-3`

E. `8`

```
public int count(String[] data){
  int result = 0;
  try{
    for(String s : data)
      result += s.length();
  }
  catch(Exception e){
    result *= -1;
  }
  return result;
}
```

QUESTION 38

What replaces **<*1>** in the code to the right to redirect to the private constructor with an argument equal to `10`?

A. `this.Structure(10)`

B. `Structure(10)`

C. `super(10)`

D. `this = Structure(10);`

E. `this(10)`

QUESTION 39

Which of the following can replace **<*2>** in the code to the right to obtain a `String` representation of `obj`?

I. `obj`
II. `obj.toString()`
III. `obj + ""`

A. I only

B. II only

C. III only

D. I and II

E. II and III

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

QUESTION 40

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

A. a binary search tree

B. a hash table

C. an array based list

D. a linked list

E. a heap

```
public class Structure<E>{

  private ArrayList<ArrayList<E>> con;
  private int size;

  public Structure(){
    <*1>;
  }

  private Structure(int cap){
    con = new ArrayList<ArrayList<E>>();
    for(int i = 0; i < cap; i++)
      con.add(new ArrayList<E>());
  }

  public void add(E obj){
    int pos = getVal(obj);
    pos = Math.abs(pos % con.size());
    if(!con.get(pos).contains(obj)){
      con.get(pos).add(obj);
      size++;
      if( size > 0.75 * con.size())
        resize();
    }
  }

  public boolean present(E obj){
    int pos = getVal(obj);
    return con.get(pos).contains(obj);
  }

  public void remove(E obj){
    int pos = getVal(obj);
    con.get(pos).remove(obj);
  }

  private int getVal(E obj){
    String s = <*2>;
    int result = 0;
    for(int i = 0; i < s.length(); i++)
      result += s.charAt(i);
    return result;
  }

  private void resize(){
    Structure<E> temp;
    temp = new Structure<E>(con.size() * 2);
    for(ArrayList<E> a : con)
      for(E obj : a)
        temp.add(obj);
    con = temp.con;
  }
}
```

No Test 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 - 2010

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

24. The `EBook` class inherits the `hashCode` method from the `Object` class. The return value of the method is unpredictable and cannot be determined at compile time simply by looking at the code.

25. The client code `equals` method calls the `equals(Object obj)` method. The `EBook` class overloads the `equals` method from the `Object class` instead of overriding it. The `equals` method in the `Object` class merely checks if the references of the two objects are the same.

26. A class with one or more abstract methods must be declared `abstract` but a class with no abstract methods may still be declared `abstract` if desired.

31. `String` concatenation is O(N) where N is the number of characters in the `String`s being concatenated. n concatenations are taking place due to the loop, and the average number of characters being concatenated is `n/2` so the method is $O(N^2)$ and the time will approximately quadruple when `n` is doubled.