University Interscholastic League

Computer Science Competition

Number 125 (Invitational A - 2011)

General Directions:

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 11001_2 and 101_2?

A. 1111_2 B. 11111_2 C. 11101_2 D. 11011_2 E. 11110_2

QUESTION 2

What is output by the code to the right?

A. `4` B. `3.9` C. `2.9`

D. `0` E. `-2`

```
int x = 3 * 2 + 2 / 5 - 15 / 6;
System.out.print(x);
```

QUESTION 3

What is output by the code to the right?

A. `0` B. `0.5` C. `9`

D. `9.0` E. `9.5`

```
double total = 0;
for(int i = 0; i < 19; i++)
  total += 0.5;
System.out.print(total);
```

QUESTION 4

What is output by the code to the right?

A. `Ritchiechie`

B. `Ritchietchie`

C. `RitchieRitc`

D. `RitchieRit`

E. `tchietchie`

```
String name = "Ritchie";
String part = name.substring(3);
System.out.print(name + part);
```

QUESTION 5

What is output by the code to the right?

A. `12` B. `6.67` C. `null`

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

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

```
Object[] jumble = {12, 6.67, "AB", 13};
System.out.print(jumble[1]);
```

QUESTION 6

What is output by the code to the right?

A. `3126` B. `3123` C. `1003`

D. `1000` E. `336`

```
double a = 3.12345678;
double b = a * 10 * 100;
int x2 = (int)b + (int)a;
System.out.print(x2);
```

QUESTION 7

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

```
!p && q
```

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

QUESTION 8

What is output by the code to the right?

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

D. `12` E. `23`

```
int x3 = 11;
if(x3 > 0)
  System.out.print(1);
if(x3 > 10)
  System.out.print(2);
if(x3 > 100)
  System.out.print(3);
```

QUESTION 9

What replaces **`<*1>`** in the code to the right so that `drinksMade` is a class variable accessible only inside the `Drink` class?

A. `static`

B. `private`

C. `private static`

D. `private static final`

E. `private class final`

```
public class Drink{
  <*1> int drinksMade;

  private double price;

  public Drink() { this(1.99); }

  public Drink(double p){
    price = p;
    drinksMade++;
  }

  public static int total(){
    return drinksMade;
  }
}

// client code
Drink d = new Drink();
d = new Drink(1.99);
System.out.print(<*2>);
```

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

QUESTION 10

Which of the following can replace **`<*2>`** in the client code to the right to call the `total` method from the `Drink` class without a syntax error?

A. `d.price` B. `Drink.total()`

C. `total` D. `d.drinksMade`

E. `Drink.price`

QUESTION 11

What is output by the code to the right?

A. `0` B. `1` C. `4`

D. `5` E. `20`

```
int total = 0;
for(int i = 0; i < 20; i++)
  if( i % 4 == 0 )
    total++;
System.out.print(total);
```

QUESTION 12

What is output by the code to the right?

A. `4` B. `4.9` C. `5.0`

D. `5` E. `10`

```
double m2 = 4.99;
System.out.print(Math.ceil(m2));
```

QUESTION 13

What is output by the code to the right?

A. `Two2     One` B. `Two2One`

C. `Two\\2One` D. `Two     2One`

E. `Two     2     One`

```
System.out.print("Two\t2One");
```

QUESTION 14

What is output by the code to the right?

A. `5472.12`

B. `5,472.1200000`

C. `05472.120`

D. `5,472.120,000,0`

E. `5,472.1,200,000`

```
System.out.printf("%,5.7f", 5472.12);
```

QUESTION 15

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

A. `3` B. `4` C. `6`

D. `8` E. `9`

```
public int process(int x, int y){
  x = y;
  x++;
  y++;
  return x * y;
}
```

QUESTION 16

What is output by the code to the right?

A. `1` B. `9` C. `27`

D. `64` E. `81`

```
String stars = "";
for(int i = 0; i < 3; i++)
  for(int j = 0; j < 3; j++)
    for(int k = 0; k < 3; k++)
      stars += "*";
System.out.println(stars.length());
```

QUESTION 17

What replaces **`<*1>`** in the code to the right so that the output is `4`?

A. `int val = 0` B. `int val = 4`

C. `int val = 20` D. `int val = 35`

E. `int val = 50`

```
<*1>;
int c = 0;
while( val >= 5 ) {
  val /= 2;
  c++;
}
System.out.println( c );
```

QUESTION 18

What is output by the code to the right?

A. `true 1` B. `false 0`

C. `true 0` D. `false 1`

E. `false false`

```
int[] list1 = new int[5];
int[] list2 = {0, 0, 0, 0, 0};
System.out.print( list1 == list2 );
System.out.print(" " + list1[1]);
```

QUESTION 19

What is output by the code to the right?

A. `12` B. `13.14`

C. `\12` D. `\t`

E. `13`

```
String st;
st = "12\n\t13.14\n\t\n\\12\t\n15";
Scanner sc = new Scanner(st);
sc.next();
sc.next();
System.out.print(sc.next());
```

QUESTION 20

What replaces **<*1>** in the code to the right to generate an exception and disrupt the normal flow of program execution if the precondition of method `myst` is not met?

A. `catch` B. `try` C. `throw`

D. `continue` E. `volatile`

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

QUESTION 21

What replaces **<*2>** in the code to the right so that the value stored in `LIMIT` may not be altered after initially assigned a value?

A. `static` B. `const`

C. `final` D. `strictfp`

E. `static final`

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

QUESTION 22

What is returned by the method call `myst(36)`?

A. 3 B. 4 C. 5

D. 9 E. 10

```
// pre: val > 0
public int myst(int val) {
  if( !(val > 0) )
    <*1> new IllegalArgumentException();

  int res = 2;
  <*2> int LIMIT = (int) Math.sqrt(val);
  for(int i = 2; i < LIMIT; i++)
    if( val % i == 0 )
      res += 2;
  if( val % LIMIT == 0 )
    res++;
  return res;
}
```

QUESTION 23

Which searching algorithm does method `search` implement?

A. `heap search` B. `sequential search`

C. `radix search` D. `stooge search`

E. `binary search`

QUESTION 24

What is returned by the method call `search(new int[0], 0)`?

A. -1 B. 0

C. 1 D. 2

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

QUESTION 25

What is output by the client code to the right?

A. 1-1 B. 2-1 C. 3-1

D. 12 E. 13

```
public int search(int[] data, int t) {
  int x = 0;
  int y = data.length - 1;
  int c = 0;
  while( x <= y ){
    c++;
    int z = (x + y) / 2;
    if( data[z] == t )
      return z;
    else if( data[z] < t )
      x = z + 1;
    else
      y = z - 1;
  }
  System.out.print(c);
  return -1;
}

// client code
int[] vs = {-1, 5, 10, 20, 30, 35};
System.out.print( search(vs, 15) );
```

QUESTION 26

Which of the following is not a syntactically correct Java identifier?

A. `_sgh` B. `bonus12` C. `LIM_DIM_` D. `bsk` E. `#CSharp`

QUESTION 27

What is output by the client code to the right?

A. `z` B. `A`

C. `a` D. `AA`

E. `Z`

QUESTION 28

Which sorting algorithm does method `sort` implement?

A. `selection sort`

B. `insertion sort`

C. `merge sort`

D. `radix sort`

E. `quicksort`

```
public void sort(String[] w){
  int lim = w.length - 1;
  for(int i = 0; i < lim; i++){
    int m = i;
    for(int j = i + 1; j <= lim; j++){
      if( w[j].compareTo(w[m]) < 0 ){
        m = j;
      }
    }
    String t = w[i];
    w[i] = w[m];
    w[m] = t;
  }
}

// client code
String[] ws = {"Z", "AA", "a", "A", "z"};
sort(ws);
System.out.print( ws[1] );
```

QUESTION 29

What is output by the code to the right?

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

D. `-2.0` E. `0`

```
System.out.print( Math.floor(-1.56) );
```

QUESTION 30

What replaces **`<*1>`** in the code to the right so the line of code marked `// A` is average case O(1) given there are N elements already present in the `Set`?

A. `HashSet<Character>`

B. `TreeSet<Character>`

C. `Set<Character>`

D. `Collection<Character>`

E. `Iterator<Character>`

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

QUESTION 31

What is output by the code to the right?

A. `17` B. `3` C. `8`

D. `16` E. `5`

```
String ds = "AABbAaAAbCAaaBBbC";
Set<Character> set;
set = new <*1>();

for(int i = 0; i < ds.length(); i++)
  set.add( ds.charAt(i) ); // A

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

QUESTION 32

A method uses the insertion sort algorithm to sort an array of `int`s. Given an array with 100,000 distinct values in random order, it takes the method 3 seconds to complete. What is the expected time for the method to complete given an array with 300,000 distinct values in random order?

A. 6 seconds B. 9 seconds C. 12 seconds D. 27 seconds E. 36 seconds

QUESTION 33

What is output by the client code to the right?

A. `7` B. `6` C. `4`

D. `3` E. `2`

QUESTION 34

What is output by the client code to the right if the inner `for` loop's initialization statement is changed from

`int j = i + 1`

to

`int j = 0`

A. `7` B. `10` C. `15`

D. `14` E. `21`

```
public int calc(int[] list) {
  int t = 0;
  int lim = list.length;
  for(int i = 0; i < lim; i++)
    for(int j = i + 1; j < lim; j++)
      if(list[i] == list[j])
        t++;
  return t;
}

// client code
int[] bd = {3, 1, 3, 1, 3, 3, 2};
System.out.print(calc(bd));
```

QUESTION 35

What kind of graph does the picture to the right represent?

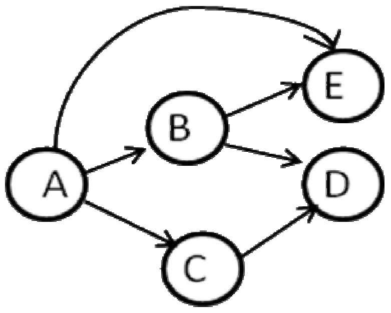
A. a directed unweighted graph

B. a directed weighted graph

C. an undirected unweighted graph

D. a undirected weighted graph

E. a binary search tree



QUESTION 36

What is returned by the method call `h(6)`?

A. `33` B. `26` C. `4`

D. `2` E. `1`

```
public int h(int x){
  if(x <= 2)
    return x * 2;
  else
    return h(x - 2) + h(x - 1) + 1;
}
```

QUESTION 37

What is output by the code to the right?

A. `12 3` B. `5 2` C. `5 1`

D. `7 2` E. `7 1`

```
List<Integer> li;
li = new LinkedList<Integer>();
li.add(5);
li.add(12);
li.add(1, 7);
int res = ((LinkedList<Integer>)
                            li).removeFirst();
System.out.print(res + " " + li.size());
```

QUESTION 38

What replaces **`<*1>`** in the code to the right to insert the `Pair p` at position `pos` in `con`?

A. `con.add(p)`

B. `con.insert(pos, p)`

C. `con.insert(p, pos)`

D. `con.add(pos, p)`

E. `con.addFirst(p)`

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

QUESTION 39

What is output by the following client code?

```
Structure s = new Structure();
s.add(12, 5);
s.add(5, 12);
s.add(17, 13);
s.add(5, 7);
System.out.print( s.remove() );
System.out.print( " " + s.remove() );
```

A. `17 12`

B. `5 5`

C. `12 7`

D. `13 5`

E. `12 5`

QUESTION 40

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

A. a binary search tree

B. a linked list

C. a priority queue

D. a stack

E. a graph

```
public class Structure {

  private List<Pair> con;

  public Structure(){
    con = new ArrayList<Pair>();
  }

  public Object get(){
    return con.get(0).value();
  }

  public Object remove(){
    return con.remove(0).value();
  }

  public void add(int x, Object obj){
    int pos = 0;
    while(pos < con.size()
              && x < con.get(pos).num() ) {
      pos++;
    }
    Pair p = new Pair(x, obj);
    <*1>;
  }

  public boolean empty(){
    return con.size() == 0;
  }

  private static class Pair {
    private int n;
    private Object obj;

    public Pair(int num, Object val){
      n = num;
      obj = val;
    }

    public int num() { return n; }

    public Object value() { return obj; }
  }
}
```

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()`
- `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.ArrayList<E> implements List<E>`

`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 Invitational A 2011

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

Notes: