UIL COMPUTER SCIENCE WRITTEN TEST

2016 INVITATIONAL B

FEBRUARY/MARCH 2016

General Directions (Please read carefully!)

1. DO NOT OPEN THE EXAM UNTIL TOLD TO DO SO.
2. There are 40 questions on this contest exam. You will have 45 minutes to complete this contest.
3. All answers must be legibly written on the answer sheet provided. Indicate your answers in the appropriate blanks provided on the answer sheet. Clean erasures are necessary for accurate grading.
4. You may write on the test packet or any additional scratch paper provided by the contest director, but NOT on the answer sheet, which is reserved for answers only.
5. All questions have ONE and only ONE correct answer. There is a 2-point penalty for all incorrect answers.
6. Tests 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 test until told to do otherwise. You may use this time to check your answers.
7. If you are in the process of actually writing an answer when the signal to stop is given, you may finish writing that answer.
8. All provided code segments are intended to be syntactically correct, unless otherwise stated. You may also assume that any undefined variables are defined as used.
9. A reference to many commonly used Java classes is provided with the test, and you may use this reference sheet during the contest. AFTER THE CONTEST BEGINS, you may detach the reference sheet from the test booklet if you wish.
10. Assume that any necessary import statements for standard Java SE packages and classes (e.g., `java.util`, `System`, etc.) are included in any programs or code segments that refer to methods from these classes and packages.
11. NO CALCULATORS of any kind may be used during this contest.

Scoring

1. Correct answers will receive **6 points**.
2. Incorrect answers will lose **2 points**.
3. Unanswered questions will neither receive nor lose any points.
4. In the event of a tie, the student with the highest percentage of attempted questions correct shall win the tie.

Copyright © 2016 University Interscholastic League

STANDARD CLASSES AND INTERFACES – SUPPLEMENTAL REFERENCE

package java.lang

```
class Object
  boolean equals(Object anotherObject)
  String toString()
  int hashCode()

interface Comparable<T>
  int compareTo(T anotherObject)
    Returns a value < 0 if this is less than anotherObject.
    Returns a value = 0 if this is equal to anotherObject.
    Returns a value > 0 if this is greater than anotherObject.

class Integer implements Comparable<Integer>
  Integer(int value)
  int intValue()
  boolean equals(Object anotherObject)
  String toString()
  String toString(int i, int radix)
  int compareTo(Integer anotherInteger)
  static int parseInt(String s)

class Double implements Comparable<Double>
  Double(double value)
  double doubleValue()
  boolean equals(Object anotherObject)
  String toString()
  int compareTo(Double anotherDouble)
  static double parseDouble(String s)

class String implements Comparable<String>
  int compareTo(String anotherString)
  boolean equals(Object anotherObject)
  int length()
  String substring(int begin)
    Returns substring(from, length()).
  String substring(int begin, int end)
    Returns the substring from index begin through index (end - 1).
  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 fromIndex. Returns -1 if str is not found.
  int indexOf(int ch)
  int indexOf(int ch, int fromIndex)
  char charAt(int index)
  String toLowerCase()
  String toUpperCase()
  String[] split(String regex)
  boolean matches(String regex)
  String replaceAll(String regex, String str)

class 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 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, int b)
  static int max(int a, int b)
  static long round(double a)
  static double random()
    Returns a double greater than or equal to 0.0 and less than 1.0.
```

package java.util

```
interface List<E>
class ArrayList<E> implements List<E>
  boolean add(E item)
  int size()
  Iterator<E> iterator()
  ListIterator<E> listIterator()
  E get(int index)
  E set(int index, E item)
  void add(int index, E item)
  E remove(int index)

class LinkedList<E> implements List<E>, Queue<E>
  void addFirst(E item)
  void addLast(E item)
  E getFirst()
  E getLast()
  E removeFirst()
  E removeLast()

class Stack<E>
  boolean isEmpty()
  E peek()
  E pop()
  E push(E item)

interface Queue<E>
class PriorityQueue<E>
  boolean add(E item)
  boolean isEmpty()
  E peek()
  E remove()

interface Set<E>
class HashSet<E> implements Set<E>
class TreeSet<E> implements Set<E>
  boolean add(E item)
  boolean contains(Object item)
  boolean remove(Object item)
  int size()
  Iterator<E> iterator()
  boolean addAll(Collection<? extends E> c)
  boolean removeAll(Collection<?> c)
  boolean retainAll(Collection<?> c)

interface Map<K,V>
class HashMap<K,V> implements Map<K,V>
class TreeMap<K,V> implements 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()

interface Iterator<E>
  boolean hasNext()
  E next()
  void remove()

interface ListIterator<E> extends Iterator<E>
  void add(E item)
  void set(E item)

class Scanner
  Scanner(InputStream source)
  Scanner(String str)
  boolean hasNext()
  boolean hasNextInt()
  boolean hasNextDouble()
  String next()
  int nextInt()
  double nextDouble()
  String nextLine()
  Scanner useDelimiter(String regex)
```

Copyright © 2016 University Interscholastic League

UIL COMPUTER SCIENCE WRITTEN TEST – 2016 INVITATIONAL B

Note: Correct responses are based on **Java SE Development Kit 8 (JDK 8)** from Sun Microsystems, Inc. All provided code segments are intended to be syntactically correct, unless otherwise stated (e.g., "error" is an answer choice) and any necessary Java SE 8 Standard Packages have been imported. Ignore any typographical errors and assume any undefined variables are defined as used. **For all output statements, assume that the System class has been statically imported using:**

```
import static java.lang.System.*;
```

Question 1.

Which of the following is equivalent to $2A_{16}$ * 3_8?

A) $3e_{16}$ **B)** 174_8 **C)** 1223_4 **D)** 1111110_2 **E)** 226_{10}

Question 2.

What is the output of the code segment to the right?

A) 7.5 **B)** 7.0 **C)** 7 **D)** 6

E) No output due to an error.

```
int x = 3;
double y = x - 0.5;
out.println(x * y);
```

Question 3.

What is the output of the code segment to the right?

A) d20.o16.x32 **B)** d20.o20.x20

C) d20.o24.x14 **D)** d%d.o%o.x%x

E) No output due to an error.

```
out.printf("d%d.o%o.x%x", 20, 20, 20);
```

Question 4.

What is the output of the code segment to the right?

A) 2 **B)** 4 **C)** 10 **D)** 11 **E)** 12

```
String dna = "ACAAGATGCCATTGTC";
int seqA = dna.indexOf("CAA");
int seqB = dna.indexOf("CA", seqA);
out.println(seqA + seqB);
```

Question 5.

Which of the following values for p, q, and r will cause the Boolean expression to the right to evaluate to true?

A) p = false; q = false; r = false;

B) p = false; q = true; r = false;

C) p = true; q = false; r = true;

D) p = true; q = true; r = false;

E) p = true; q = true; r = true;

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

Question 6.

What is the output of the code segment to the right?

A) 3 **B)** 3.0 **C)** 3.14 **D)** 4

E) No output due to an error.

```
int appxPi = Math.round(Math.PI);
out.println(appxPi);
```

Question 7.

What is the output of the code segment to the right?

A) 0.2 **B)** 5 **C)** 9 **D)** 20 **E)** 25

E) No output due to an error.

```
int val = 45;
out.println(val %= 20);
```

Question 8.

What is the output of the code segment to the right if the value of num is initialized as follows?

```
int num = 28;
```

A) WY **B)** W **C)** X

D) Y **E)** Z

```
if ((num % 3 == 0) || (num % 4 == 0))
  out.print("W");
else if (num % 3 == 0)
  out.print("X");
else if (num % 2 == 0)
  out.print("Y");
else
  out.print("Z");
```

Copyright © 2016 University Interscholastic League

Question 9.

What is the output of the code segment to the right?

A) ++++++++ **B)** +++++++

C) ++++++ **D)** +++++

E) No output due to an infinite loop.

```
int control = 64;
while (control > 1) {
  control /= 2;
  if (control % 2 == 0)
    out.print("+");
}
```

Question 10.

What is the output of the code segment to the right?

A) [4, 5, 6, 7, 8] **B)** [4, 5, 4, 3, 2]

C) [4, 4, 3, 2, 1] **D)** [4, 3, 2, 1, 0]

E) No output due to an error.

```
int[] digits = {4, 3, 2, 1, 0};
for (int i = 0; i < 4; i++)
  digits[i + 1] = digits[i] + 1;
out.println(Arrays.toString(digits));
```

Question 11.

Assuming that `data.txt` contains multiple lines of space-separated integers, similar to what is shown to the right, which of the following could replace **<#1>** and **<#2>** in this code segment?

	<#1>	**<#2>**
A)	`fin.hasNext()`	`fin.next()`
B)	`fin.hasNextLine()`	`fin.nextLine()`
C)	`fin.nextInt()`	`fin.nextInt()`
D)	`fin.nextInt()`	`fin.hasNextInt()`
E)	`fin.hasNextInt()`	`fin.nextInt()`

data.txt

```
11   9  70
 3 -50  19
12   5   7
 1   4  -3
```

```
int sum = 0;
File file = new File("data.txt");
Scanner fin = new Scanner(file);
while (<#1>)
  sum += <#2>;
out.println(sum);
```

Question 12.

Assuming that **<#1>** and **<#2>** have been filled in correctly and the `data.txt` file contains the values shown to the right, what is the output of this code segment?

A) 2 **B)** 27 **C)** 88 **D)** 90 **E)** 93

Question 13.

What is the output of the code segment to the right?

A) 2.0 **B)** 6.0 **C)** 14.0 **D)** 15.5 **E)** 18.0

```
double dbl = 5.0 + 7 / 2 * 3;
out.println(dbl);
```

Question 14.

Which of the following data types CANNOT be assigned to a `float` variable without encountering a possible loss of precision?

A) double **B)** char **C)** int **D)** byte **E)** short

Question 15.

What is the output of the code segment to the right?

A) aage's caages **B)** abage's cabages

C) segaac s'egaa **D)** segabac s'egaba

E) No output due to an error.

```
String words = "babbage's cabbages";
List<Character> letters = new ArrayList<>();
for (int i = 0; i < words.length(); i++)
  letters.add(0, words.charAt(i));
for (int i = 0; i < letters.size(); i++)
{
  if (letters.get(i) == 'b')
    letters.remove(i);
}
words = "";
for (char letter : letters)
  words += letter;
out.println(words);
```

Copyright © 2016 University Interscholastic League

Question 16.

Which of the following regular expressions could NOT replace **<#1>** in the method to the right so that `toInches()` properly returns the total number of inches represented by the formatted `height` parameters as shown in the sample **Client Code**?

A) `\\D+`

B) `[int f]+`

C) `\\D\\D\\W?`

D) `(ft )|(in)`

E) `ft|in`

```
public static int toInches(String height) {
  String[] ftIn = height.split("<#1>");
  int feet = Integer.parseInt(ftIn[0]);
  int inch = Integer.parseInt(ftIn[1]);
  return (feet * 12) + inch;
}
```

Client Code

```
int x = toInches("5ft 8in");
int y = toInches("10ft 0in");
int z = toInches("100ft 11in");
```

Question 17.

What is the output of the code segment to the right?

A) `true` **B)** `false` **C)** `yes` **D)** `no`

E) No output due to an error.

```
String ans = -0.0 < 0.0 ? "yes" : "no";
out.println(ans);
```

Question 18.

Given the `notNice()` method to the right, what is the output of the following client code?

```
int[] scores = {7, 3, 8, 0, -2};
out.println(notNice(scores));
```

A) `0.0` **B)** `2.0` **C)** `2.2` **D)** `3.0` **E)** `3.2`

```
static double notNice(int[] a) {
  int t = 0;
  for (int i : a)
   t += i;
  return t / a.length;
}
```

Question 19.

What is the output of the code segment to the right?

A) `100` **B)** `107` **C)** `128` **D)** `137` **E)** `150`

```
int sum = 0;
for (int i = 0; i < 5; i++) {
  for (int j = i; j < 10; j += 2) {
    sum += j;
  }
}
out.println(sum);
```

Question 20.

What is the output of the code segment to the right?

A) `[13, 9, 3, 11, 5, 12]`

B) `[3, 5, 9, 11, 12, 13]`

C) `[3, 5, 9, 13, 11, 12]`

D) `[13, 11, 12, 9, 5, 3]`

E) `[13, 12, 11, 9, 5, 3]`

```
Queue<Integer> pq = new PriorityQueue<>();
pq.add(13);
pq.add(9);
pq.add(3);
pq.add(11);
pq.add(5);
pq.add(12);
out.println(pq);
```

Question 21.

What is the output of the code segment to the right?

A) `[blackjack, Two One, 21, 10101]`

B) `[21, 10101, Two One, blackjack]`

C) `[21, Two One, 10101, blackjack]`

D) `[10101, 21, Two One, blackjack]`

E) `[Two One, blackjack, 10101, 21]`

```
String[] xxi = { "21", "Two One", "10101",
                 "blackjack"};
for (int i = 0; i < xxi.length; i++) {
  for (int j = 0; j < xxi.length - 1; j++) {
    int n = xxi[j].compareTo(xxi[j+1]);
    if (n < 0) {
      String t = xxi[j];
      xxi[j] = xxi[j+1];
      xxi[j+1] = t;
    }
  }
}
out.println(Arrays.toString(xxi));
```

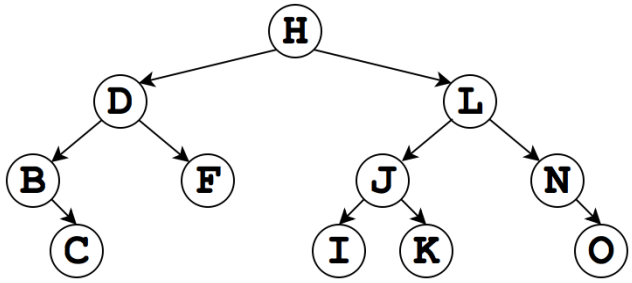
Copyright © 2016 University Interscholastic League

Question 22.

Considering the tree to the right, which of the following reflects the order in which the nodes are first encountered in a Depth-First Search (DFS)?

A) BCDFHIJKLNO **B)** HDLBFJNCIKO **C)** HDBCFLJIKNO

D) CIKOBFJNDLH **E)** CBFDIKJONLH



Question 23.

Considering the tree to the right, which of the following reflects the order in which the nodes are first encountered in a Breadth-First Search (BFS)?

A) BCDFHIJKLNO **B)** HDLBFJNCIKO **C)** HDBCFLJIKNO

D) CIKOBFJNDLH **E)** CBFDIKJONLH

Question 24.

What is the output of the code segment to the right?

A) 7 **B)** 9 **C)** 143 **D)** 150 **E)** 2025

```
out.println(135 & 15);
```

Question 25.

Which of the following could replace **<#1>** in the Omega class to the right?

A) this("1"); **B)** super(1); **C)** Alpha("1");

D) super("1"); **E)** More than one of these.

Question 26.

Assuming that **<#1>** has been completed correctly, what is the output of the **Client Code #1** to the right?

A) 1-0 3-0

B) 3-0 3-0

C) 1-1 3-3

D) 3-3 3-3

E) 2-0 2-0

Question 27.

Assuming that **<#1>** has been completed correctly, what is the output of the **Client Code #2** to the right?

A) 0

B) 1

C) 2

D) 3

E) 4

Question 28.

Which of the following would be a valid client code statement?

A) Alpha a = new Alpha("5");

B) Alpha b = new Inty();

C) Omega c = new Omega();

D) Inty d = new Alpha("6");

E) Inty e = new Inty(7);

```
interface Inty { public int toInt(); }

abstract class Alpha {
  private String data = "0";
  private static int inty = 0;

  public Alpha(String d) { data = d; }

  public String toString() {
    return "" + inty;
  }
}

class Omega extends Alpha implements Inty {
  private String data = "2";
  private static int inty = 2;

  public Omega() { this(2); }

  public Omega(int i) {
    <#1>
    inty = i;
  }

  public String toString() {
    return toInt() + "-" + super.toString();
  }

  public int toInt() { return inty; }
}
```

Client Code #1

```
Alpha beta = new Omega(1);
Alpha gamma = new Omega(3);
out.print(beta + " ");
out.print(gamma);
```

Client Code #2

```
Inty bravo = new Omega(4);
out.print(bravo.toInt());
```

Copyright © 2016 University Interscholastic League

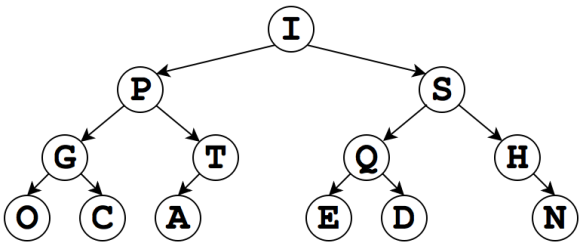
Question 29.

Assuming that `root` references the root node of the tree shown to the right and that each of the nodes in the tree is represented by an instance of the `Node` class, what value is returned by the following Client Code invocation of the `huff()` method?

```
huff(root, 2658);
```

A) ACED **B)** DECADE **C)** DECA

D) ACES **E)** No output due to an error.



Question 30.

Which of the following conditions can cause an invocation of the `huff()` method to lead to a runtime error due to infinite recursion?

A) `c == 0` **B)** `r` references an empty tree

C) `r == null` **D)** `r` references a leaf node

E) The methods will never lead to infinite recursion.

Node.java

```
public class Node {
  public String value;
  public Node left;
  public Node right;
}
```

Question 31.

Which of the following conditions will always cause an invocation of the `huff()` method to lead to a runtime exception?

A) `c > 0` **B)** `c < 0`

C) `r == null` **D)** `r` references a leaf node

E) The methods will never result in a runtime exception.

```
public String huff(Node r, int c) {
  return help(r, r, c);
}

private String help(Node r, Node n, int c) {
  if (c == 0) return n.value;
  if (c % 2 == 0 && n.left != null)
    return help(r, n.left, c/2);
  else if (c % 2 != 0 && n.right != null)
    return help(r, n.right, c/2);
  else
    return n.value + huff(r, c);
}
```

Question 32.

What is the output of the code segment to the right?

A) `[M, a, s]`

B) `[{T:M}, {r:a}, {e:p}, {e:s}]`

C) `[e, e, r, T]`

D) `[a, M, p, s]`

E) `[T, e, r]`

```
Map<String, String> map = new TreeMap<>();
map.put("T", "M");
map.put("r", "a");
map.put("e", "p");
map.put("e", "s");
out.println(map.keySet());
```

Question 33.

What value does the postfix expression to the right evaluate to?

A) −79 **B)** 25 **C)** 55 **D)** 64 **E)** 240

```
4 8 7 + 2 * 3 1 - * +
```

Question 34.

What type of data structure does the tree to the right represent?

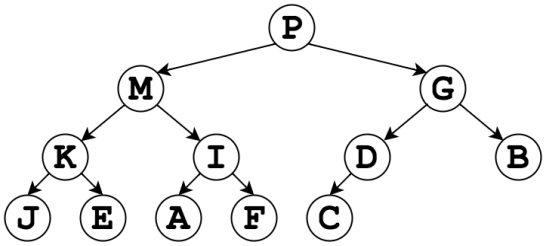
A) Min-heap

B) Max-heap

C) Binary Search Tree

D) A and C only

E) B and C only



Question 35.

Which of the following is equivalent to the Boolean expression shown to the right?

A) $X\overline{Y} + XZ$ **B)** $\overline{X}Y + \overline{X}\,\overline{Z}$ **C)** $X + \overline{Y}Z$

D) $X(\overline{Y} + Z)$ **E)** $X + (\overline{Y} + Z)$

$$(X + \overline{Y})(X + Z)$$

Copyright © 2016 University Interscholastic League

Question 36.

What type of data structure is modeled by the `Struct` class defined to the right?

A) A graph

B) A priority queue

C) A binary tree

D) A linked list

E) A stack

```
class Struct implements Comparable<Struct> {
  static Set<Struct> all = new TreeSet<>();
  private Set<Struct> neighbors;

  public Struct() {
    neighbors = new TreeSet<Struct>();
    all.add(this);
  }

  public Struct(Struct x) {
    this();
    x.add(this);
  }

  public boolean isAdjacent(Struct x) {
    return neighbors.contains(x);
  }

  public boolean add(Struct x) {
    neighbors.add(x);
    return x.isAdjacent(this);
  }

  public int compareTo(Struct x) {
    return this.hashCode() - x.hashCode();
  }
}
```

Question 37.

What is the expected runtime performance for the `Struct` class' `isAdjacent()` method in the worst case? Choose the most restrictive answer.

A) $O(1)$ **B)** $O(N)$ **C)** $O(N^2)$

D) $O(\log_2 N)$ **E)** $O(N * \log_2 N)$

Question 38.

What is the output of the **Client Code** segment to the right?

A)
```
false
true
false
true
```

B)
```
false
false
false
true
```

C)
```
true
false
true
true
```

D)
```
true
true
false
true
```

E)
```
false
false
true
true
```

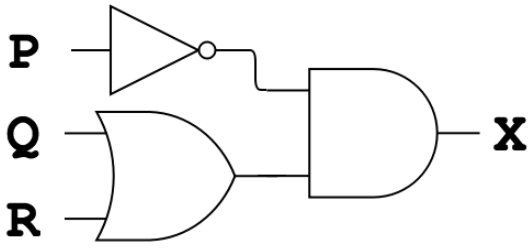
Client Code

```
Struct v1 = new Struct();
Struct v2 = new Struct(v1);
Struct v3 = new Struct();
Struct v4 = new Struct(v2);

v1.add(v2);
v1.add(v4);
v2.add(v3);

out.println(v2.isAdjacent(v1));
out.println(v3.isAdjacent(v2));
out.println(v3.add(v2));
out.println(v2.isAdjacent(v3));
```

Question 39.

What is the Boolean expression for output `X` described by the logic diagram to the right? Your answer should use the fewest logical operators as is necessary for this component.

Write your answer on the answer sheet.



Question 40.

What is the decimal equivalent to the 8-bit, 2's complement binary representation to the right?

11010110_2

Write your answer on the answer sheet.

★ DOUBLE-CHECK YOUR ANSWERS ★

Copyright © 2016 University Interscholastic League