UIL COMPUTER SCIENCE WRITTEN TEST

2016 INVITATIONAL A

JANUARY/FEBRUARY 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.

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.
  char 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)
  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)
```

UIL Computer Science Written Test – 2016 Invitational A

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 $123_8 + 45_8$?

A) 1230_4 **B)** 168_8 **C)** 78_{16} **D)** 130_{10} **E)** More than one of these

Question 2.

What is the value of z in the code segment to the right?

A) 0.0 **B)** 2.0 **C)** 3.0 **D)** 6.0

E) No output due to an error.

```
double x = 6;
double y = 1/2;
double z = x * y;
```

Question 3.

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

A) 6 Points

B) 6 Points right

C) 6 pts right

D) 6 pts

E) No output due to an error.

```
int right = 6;
int wrong = -2;
int skip = 0;
String pts = "Points";
out.printf("%d pts", right);
```

Question 4.

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

A) ihgfedc....d.f.hi **B)** ih.f.d.....d.f.hi

C) ihgfedcbabcdefghi **D)** ih.f.d....cdefghi

E) No output due to an error.

```
String pal = "ihgfedcbabcdefghi";
pal = pal.replaceAll("[cabbage]", ".");
out.println(pal);
```

Question 5.

Given the code segment to the right, when will r be false?

A) Only when a < b **B)** Never

C) Only when a > b **D)** Always

E) It is impossible to determine.

```
boolean p = (a < b);
boolean q = (a != b);
boolean r = !(p && q);
```

Question 6.

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

A) 4.0 3.0 3.0 **B)** 3.0 4.0 3.0 **C)** 3.0 4.0 4.0

D) 4.0 3.0 4.0 **E)** 3.0 3.0 3.0

```
double bang = Math.ceil(Math.PI);
double pow = Math.floor(Math.PI);
double oof = Math.min(bang, pow);
out.println(bang + " " + pow + " " + oof);
```

Question 7.

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

A) 0 **B)** 4 **C)** 8 **D)** 16

E) No output due to an error.

```
byte nibble = 4;
nibble += nibble;
nibble -= nibble;
nibble *= nibble;
nibble /= nibble;
out.println(nibble);
```

Question 8.

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

A) 1 **B)** 2 **C)** 3 **D)** 12 **E)** 13

```
int ulo = 3;
if (4 % ulo == 1)
  out.print("1");
if (5 % ulo == 2)
  out.print("2");
else
  out.print("3");
```

Question 9.

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

A) 707274 **B)** FHJL **C)** FHJ

D) FGHIJKL **E)** FGHIJK

```
for (char ch = 'F'; ch < 'L'; ch += 2)
   out.print(ch);
```

Question 10.

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

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

B) [13, 8, 3, 12, 6, 11]

C) [13, 3, 0, -1, 1, 2]

D) [13, 12, 3, 2, -1, 0]

E) No output due to an error.

```
int[] alpha = new int[6];
alpha[0] = 13;
alpha[3] = alpha[0] - 1;
alpha[2] = alpha[3] / 4;
alpha[5] = alpha[3]--;
alpha[1] = alpha[5] - alpha[2];
alpha[4] = alpha[3] / 2;
out.println(Arrays.toString(alpha));
```

Question 11.

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

A) 8 **B)** 10 **C)** 5 **D)** 28

E) No output due to an error.

```
int total = 0;
String msg = "4 -10 12 8 -6 7 3 -1 2 9 0";
Scanner parser = new Scanner(msg);
while (parser.nextInt() % 2 == 0)
  total += parser.nextInt();
out.println(total);
```

Question 12.

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

A) 31 **B)** 33 **C)** 54 **D)** 63

E) No output due to an infinite loop.

```
int seqA = 0;
int seqB = 1;
int seqSum = seqA + seqB;
while (seqSum < 50) {
  seqA = seqB;
  seqB = seqA + seqB;
  seqSum += seqB;
}
out.println(seqSum);
```

Question 13.

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

A) 2 2 2 **B)** 4 2 6 **C)** 2 3 7 **D)** 4 3 7

E) No output due to an error.

```
int q = 0;
int r = q++ + ++q;
int s = ++q + q++;
out.println(q + " " + r + " " + s);
```

Question 14.

Which pair of Java primitive data types occupies the same number bits of storage in memory?

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

Question 15.

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

A) [Donald, Huey, Dewey, Louis, Daffy]

B) [Dewey, Daffy, Louis]

C) [Dewey, Daffy, Huey, Louis]

D) [Dewey, Donald, Louis, Daffy]

E) [Dewey, Daffy, Louis, Huey]

```
List<String> ducks = new LinkedList<>();
ducks.add("Donald");
ducks.add(1, "Huey");
ducks.add(0, "Dewey");
ducks.add(ducks.size() - 1, "Louis");
ducks.set(1, "Daffy");
out.println(ducks);
```

Question 16.

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

A) 0 **B)** 56 **C)** 63 **D)** 170 **E)** 508

```
out.println(511 >> 3);
```

Question 17.

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

A) 1212210202 **B)** 000999999

C) 1212210202000 **D)** 333333

E) No output due to an error.

```
out.println(Integer.toString(999999, 3));
```

Question 18.

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

A) `[10, 10, 9, 7]`

B) `[6, 9, 9, 6]`

C) `[6, 6, 5, 3]`

D) `[10, 14, 15, 13]`

E) No output due to an error.

```
int[][] table = new int[4][5];
int[] rows = new int[table.length];

for (int i = 0; i < table.length; i++)
  for (int j = i; j < table[i].length; j++)
    table[i][j] = i + j;

int i = 0;
for (int[] row : table) {
  int tot = 0;
  for (int n : row)
    tot += n;
  rows[i++] = tot;
}
out.println(Arrays.toString(rows));
```

Question 19.

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

A) `[24k, cop, Doc, LBJ, null]`

B) `[24k, Doc, LBJ, cop, null]`

C) `[, 24k, Doc, LBJ, cop, null]`

D) `[null, , 24k, Doc, LBJ, cop]`

E) No output due to an error.

```
String[] ids = new String[3];
ids = new String[] { "LBJ", "cop", "Doc",
                     "null", "", "24k"};
Arrays.sort(ids);
out.println(Arrays.toString(ids));
```

Question 20.

Given the class definitions to the right, which of the following client code statements would be a valid variable initialization?

A) `Alpha aaa = new Alpha("Alfa");`

B) `Beta bbb = new Beta("Bravo");`

C) `Alpha ccc = new Beta();`

D) `Beta ddd = new Alpha();`

E) More than one of the above.

Question 21.

Given the class definitions to the right, what is the output of the following client code segment?

```
Alpha agent = new Beta("007");
out.println(agent.id());
```

A) `007` **B)** `Alpha` **C)** `Beta` **D)** `id` **E)** `Echo`

Question 22.

Given the class definitions to the right, what is the output of the following client code segment?

```
Alpha agent = new Beta("007");
out.println(agent);
```

A) `007` **B)** `Alpha` **C)** `Beta` **D)** `id` **E)** `Echo`

```
public class Alpha {
  private String id;

  public Alpha () {
    id = "Echo";
  }

  public String id() { return id; }

  public String toString() { return id; }
}

public class Beta extends Alpha {
  private String id;

  public Beta(String id) {
    this.id = id;
  }

  public String id() { return id; }
}
```

Question 23.

Which of the following Big-O approximations for an algorithm would represent the most optimal performance?

A) $O(N^3)$ **B)** $O(N)$ **C)** $O(N * \log_2 N)$ **D)** $O(\log_2 N)$ **E)** $O(N^2)$

Question 24.

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

A) `true` **B)** `false` **C)** `valid` **D)** `invalid`

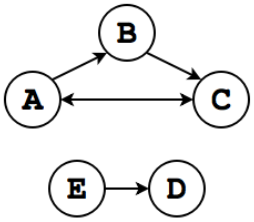
E) No output due to an error.

```
String userid = "uil_2016";
String format = "\\w+\\d+";
if (userid.matches(format))
  out.println("valid");
else
  out.println("invalid");
```

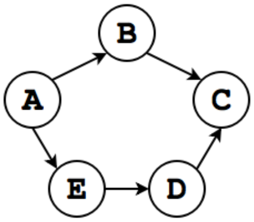
Question 25.

Which of the following graphs illustrates the connections shown in the adjacency matrix to the right?

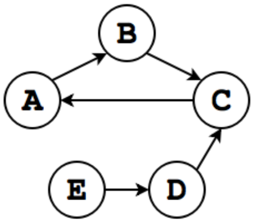
A)



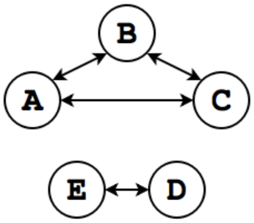
B)



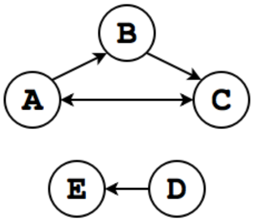
C)



D)



E)



	A	**B**	**C**	**D**	**E**
A	`false`	`true`	`true`	`false`	`false`
B	`false`	`false`	`true`	`false`	`false`
C	`true`	`false`	`false`	`false`	`false`
D	`false`	`false`	`false`	`false`	`false`
E	`false`	`false`	`false`	`true`	`false`

Question 26.

Which of the following types of graphs does the matrix to the right describe?

A) Connected graph **B)** Weighted graph

C) Directed graph **D)** A and B only

E) A and C only

Question 27.

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

A) `P || Q`

B) `!P && !Q`

C) `!P || Q`

D) `true`

E) `false`

```
!(P && Q) || Q
```

Question 28.

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

A) `fifteen` **B)** `15` **C)** `510`

D) No output due to a syntax error.

E) No output due to a runtime error.

```
out.println(Integer.parseInt(5 + "10"));
```

Question 29.

Which of the following standard searching and sorting algorithms always performs with the same O(N * $\log_2$ N) performance in the best-, average-, and worst-case scenarios?

A) Quicksort **B)** Sequential Search **C)** Merge Sort **D)** Binary Search **E)** Selection Sort

Question 30.

Given the code segment to the right, what are the contents of the `parts` array?

A) `[UIL.C, mp, t, r.Sc, , nc]`

B) `[UIL.C, mp, t, r.Sc, nc]`

C) `[L.C, mp, t, r.Sc, , nc]`

D) `[UIL.Computer.Science]`

E) `[, , L.C, mp, t, r.Sc, , nc]`

```
String whole = "UIL.Computer.Science";
String[] parts = whole.split("[aeiou]");
```

Question 31.

Given the recursive method to the right, what value is returned by invoking `sputter(8)`?

A) `#........`

B) `#........#....#..#.`

C) `#.#.#.#.#.#.#.#.`

D) `#.#..#....#........`

E) No output due to an error.

```
public static String sputter(int n) {
  if (n <= 0) { return ""; }

  String splat = "#";

  for (int i = 0; i < n; i++)
     splat += ".";

  return splat + sputter(n / 2);
}
```

Question 32.

What is the output of **Line #1** in the code segment to the right?

A) `[9, 7]` **B)** `[36, 7]`

C) `[10, 60, 11, -2]` **D)** `[10, 11]`

E) No output due to a runtime error.

Question 33.

What is the output of **Line #2** in the code segment to the right?

A) `[34, 25]` **B)** `[-14, 2]`

C) `[60, -2]` **D)** `[60, 2]`

E) No output due to an error.

```
Stack<Integer> stack = new Stack<>();
Queue<Integer> queue = new LinkedList<>();

stack.push(10);
stack.push(24);
stack.push(36);
queue.add(stack.pop() + stack.pop());
stack.push(11);
stack.push(9);
stack.push(7);
queue.add(stack.pop() - stack.pop());

out.println(stack);                          // Line #1
out.println(queue);                          // Line #2
```

Question 34.

Which of the following Boolean expressions corresponds to the logic diagram to the right?

A) $X = (P + \overline{Q}) * R$

B) $X = (\overline{P} * Q) + \overline{R}$

C) $X = (P * \overline{Q}) + R$

D) $X = (\overline{P + Q}) * \overline{R}$

E) $X = (\overline{P} + Q) * \overline{R}$

Question 35.

Which of the following set of inputs for the logic diagram to the right will result in a `true` output for `X`?

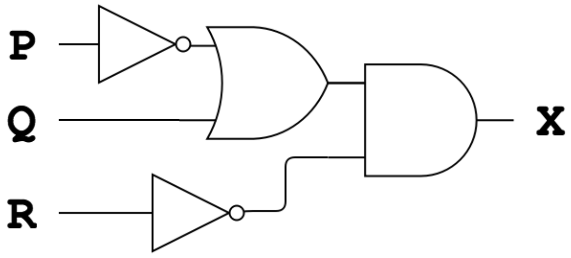
A) `P = true; Q = true; R = false;`

B) `P = false; Q = false; R = true;`

C) `P = false; Q = true; R = true;`

D) `P = true; Q = false; R = false;`

E) `P = true; Q = false; R = true;`



Question 36.

What is the 8-bit, 2's complement binary representation of -55?

A) `-00110111` **B)** `10110111` **C)** `11001000` **D)** `11001001` **E)** `-11001000`

Question 37.

What is the postfix notation for the arithmetic expression to the right?

A) `+ * z - y w x` **B)** `w x - y * z +`

C) `+ * - w x y z` **D)** `w x y z - * +`

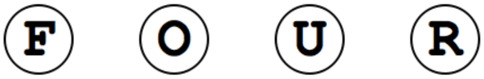
E) `z + y * (x - w)`

```
(w - x) * y + z
```

Question 38.

How many uniquely different ways can the 4 nodes to the right be arranged such that they form a valid binary search tree?

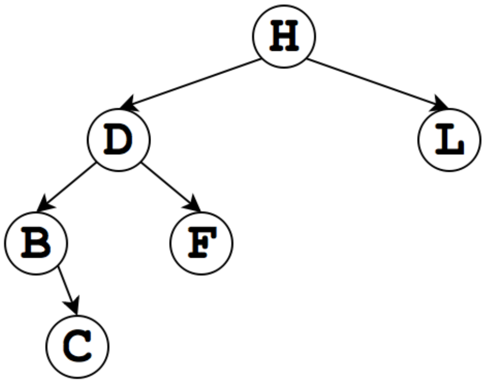
A) 1 **B)** 4 **C)** 10 **D)** 14 **E)** 24



Question 39.

What is the pre-order traversal of the nodes in the binary tree shown to the right?

Write your answer on the answer sheet.



Question 40.

Write a simplified, Boolean expression to describe output `X`, given inputs `A`, `B`, and `C`, as shown in the truth table to the right, where `0` denotes `false` and `1` denotes `true`. Your answer should use as few logical operators as possible.

Write your answer on the answer sheet.

A	B	C	X
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1