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`

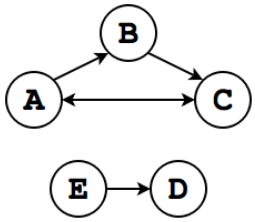
E) No output due to an error.

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

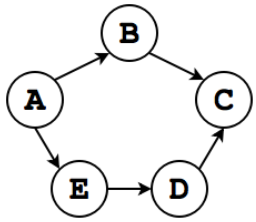
Question 25.

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

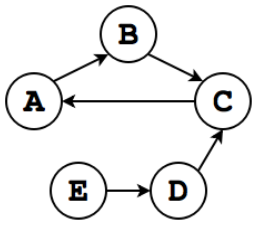
A)



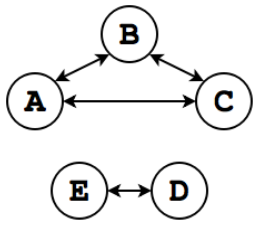
B)



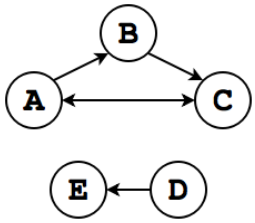
C)



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?

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

A) `P || Q`

B) `!P && !Q`

C) `!P || Q`

D) `true`

E) `false`

Question 28.

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

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

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

D) No output due to a syntax error.

E) No output due to a runtime error.

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?

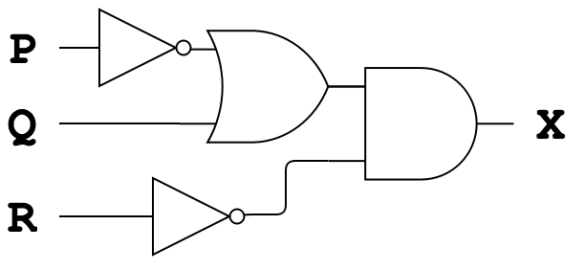
A) $X = (P + \overline{Q}) * R$

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?

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

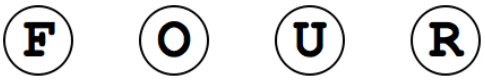
A) `+ * z - y w x` **B)** `w x - y * z +`

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

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

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?

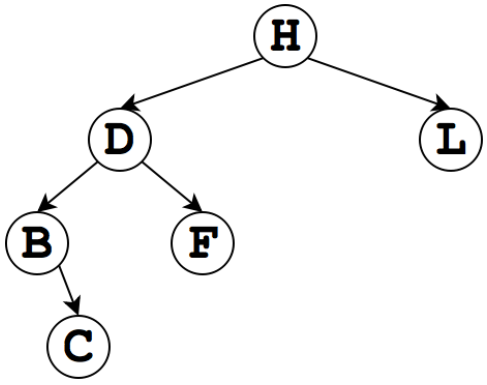


A) 1 **B)** 4 **C)** 10 **D)** 14 **E)** 24

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

Conference _____ Contestant Number _____

UIL COMPUTER SCIENCE WRITTEN TEST – 2016 INVITATIONAL A

Questions (+6 points for each correct answer, -2 points for each incorrect answer)

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

FOR ADMINISTRATIVE USE ONLY

# Right:	×	6 pts	=	
# Wrong:	×	-2 pts	=	
# Skipped:	×	0 pts	=	0

	Score	Initials
Judge #1:		
Judge #2:		
Judge #3:		

★ANSWER KEY – CONFIDENTIAL★

UIL COMPUTER SCIENCE WRITTEN TEST – 2016 INVITATIONAL A

Questions (+6 points for each correct answer, -2 points for each incorrect answer)

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

** See "Explanation" section below for alternate, acceptable answers.*

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.

Explanation

1) C `123`$_8$ `+ 45`$_8$ `= 78`$_{16}$ `= 170`$_8$ `= 1320`$_4$ `= 120`$_{10}$. Note that answer B (`168`$_8$) is not a valid octal representation.
2) A `x = 6.0`, `y = 0.0` (integer division), `z = 6 * 0 = 0.0`
3) D `"%d pts"` formats a string consisting of a decimal representation of `right` (6) followed by the string literal `" pts"`. Variables `wrong`, `skip`, and `pts` are not used in the output.
4) B `"[cabbage]"`: Regex matches on any `'a'`, `'b'`, `'c'`, `'e'`, or `'g'` (redundant `'a'` and `'b'` in regex are ignored.
5) A `a < b` → `r = false`, `a == b` → `r = true`; `a > b` → `r = true`
6) A `bang = 4.0` (`ceil()` rounds up), `pow = 3.0` (`floor()` rounds down), `oof = 3.0` (`min()` returns the minimum parameter)
7) E `nibble -= nibble` results in `nibble` being assigned a value of `0`. This later leads to `nibble /= nibble`, which results in a "Divide by Zero" exception.
8) D Both `if()` conditions are independent of one another, so the 2nd `if()` will always be evaluated, regardless of the outcome of the 1st `if()`. The else clause is not evaluated here since it is contingent upon the 2nd `if()` evaluating to false, which it doesn't.
9) C `ch` iterates through every other uppercase letter from `'F'` through `'L'` (exclusive). Each character literal is printed in succession, all on the same line of output.

★ANSWER KEY – CONFIDENTIAL★

10) A `alpha[5] = alpha[3]--;`: The post-decrementation operator (`--`) assigns the original value of `alpha[3]` (e.g., 12) to `alpha[5]`. Then `alpha[3]` is decremented from 12 to 11.

11) C `total = -10 + 8 + 7 = 5`: Each iteration of the loop reads of 2 integers from the input string. If the first integer is even, the second integer is added to `total`. The loop exits when the `Scanner` reads off the 7.

12) D This code accumulates powers of 2. At the end of each iteration, `seqA` and `seqB` are always powers of 2. The loop exits when the accumulated sum reaches 63.

13) B `r = 0 + 2 = 2` (side effect: `q = 2`). `s = 2 + 4 = 6` (side effect: `q = 4`).

14) C `short` and `char` = 16 bits of memory

15) E `"Dewey"` inserts before `"Donald"`, `"Louis"` inserts before `"Huey"`, `"Daffy"` replaces `"Donald"`

16) C 511_{10} = 0111111111_2 >> 3 = 0000111111_2 = 63_{10}. Also `511 / 2^3 = 511 / 8 = 63.875` → truncates to 63.

17) C 999999_{10} = 1212210202000_3

18) D `table:`

0	**1**	**2**	**3**	**4**
1	**2**	**3**	**4**	**5**
2	3	**4**	**5**	**6**
3	4	5	**6**	**7**

`0 + 1 + 2 + 3 + 4 =` **10**
`2 + 3 + 4 + 5 =` **14**
`4 + 5 + 6 =` **15**
`6 + 7 =` **13**

`rows:`

10
14
15
13

19) C Strings are sorted lexicographically by Unicode character values. `"null"` is a `String` literal, not the `null` reference.

20) B `Alpha` is abstract and cannot be instantiated as a `new Alpha()`. Constructor `Beta(String id)` in class `Beta` requires a `String` parameter and cannot be instantiated as a `new Beta()`.

21) A Since `agent` is instantiated as a `new Beta("007")`, the overridden method `id()` in the `Beta` class is used, which accesses the instance variable `id` privately declared within the `Beta` class (i.e., initialized w/ `"007"`).

22) E The `toString()` method is inherited from the `Alpha` class, which accesses the instance variable `id` privately declared within the `Alpha` class (i.e., initialized w/ `"Echo"`).

23) D $O(1) < O(\log_2 N) < O(N) < O(N * \log_2 N) < O(N^2)$

24) C Regular expression `"\w+\d+"` (encoded as a String w/ escape characters: `"\\w+\\d"`) means "one or more word characters (i.e., `\w` = `[a-zA-Z_0-9]` followed by one or more digit characters (i.e., `\d` = `[0-9]`)".

25) A Each true value in the matrix correspond to a directed link from the node for the column to the node for the row.

26) C Links are represented by non-weighted Boolean values. The matrix is not symmetric (i.e., a directed graph). Nodes `D` and `E` are disconnected from nodes `A`, `B`, and `C`.

27) D

P	Q	**!(P && Q) || Q**	P || Q	!P && !Q	!P || Q
0	0	**1**	0	1	1
0	1	**1**	1	0	1
1	0	**1**	1	0	0
1	1	**1**	1	0	1

28) C String concatenation results in the `String "510"` to be parsed into an integer.

29) C *Quicksort*: $O(N^2)$ in worst case. *Sequential Search*: O(N) in worst and average cases, O(1) in best case. *Binary Search* $O(\log_2 N)$ in worst and average cases, O(1) in best case. *Selection Sort*: $O(N^2)$ in all cases.

30) A String splits on all individual lowercase vowels. Empty string at the end of the array after the final "e" is truncated.

31) B Each recursive pass appends a "splat" (#) and `n` trailing dots to the end of the resulting string. Base case: `n <= 0`.

32) D `stack = [10, ~~24~~, ~~36~~, 11, ~~9~~, ~~7~~] = [10, 11]`

33) C `queue = [(36 + 24), (7 - 9)] = [60, -2]`

34) E Java equivalent: `(!P || Q) && !R`

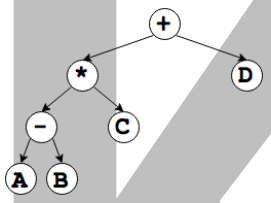
★ANSWER KEY – CONFIDENTIAL★

35) A

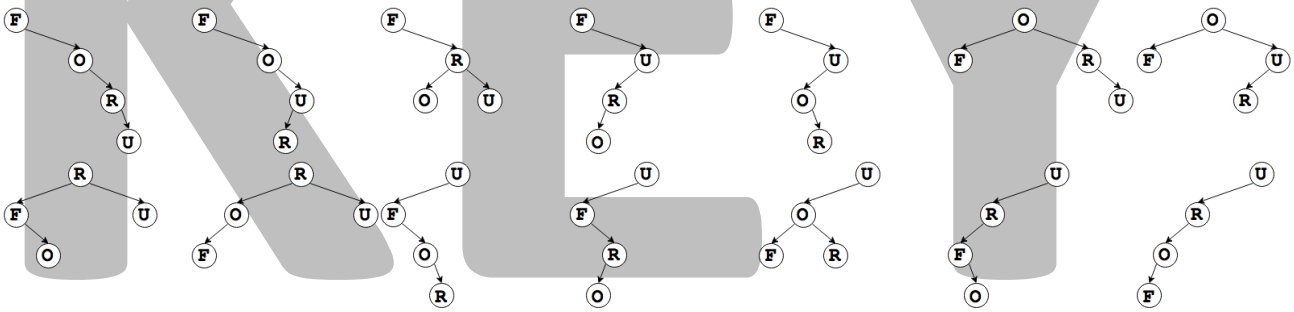
P	Q	R	$(\overline{P} + Q) \bullet \overline{R}$
0	0	0	**1**
0	0	1	**0**
0	1	0	**1**
0	1	1	**0**
1	0	0	**0**
1	0	1	**0**
1	1	0	**1**
1	1	1	**0**

36) D $55_{10} = 00110111_2$

37) B



38) D



39) Pre-order: `HDBCFL`. Post-order: `CBFDLH`. In-order: `BCDFHL`. Level-by-level: `HDLBFC`.

40) Any answer that equivalently expresses "A Logical-OR C" is acceptable (e.g., `"A + C"`, `"C + A"`, `"A || C"`, `"C || A"`, `"A or C"`, `"C or A"`)

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.

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)
```

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");
```

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);
```

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]`

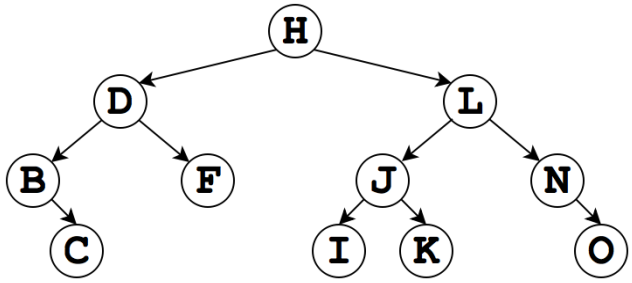
```
String[] xxi = { "21", "Two One", "10101",
                 "blackjack"};
for (int i = 0; i < xxi.length; i++) {
  for (int j = 0; j < xxi.length - 1; j++) {
    int n = xxi[j].compareTo(xxi[j+1]);
    if (n < 0) {
      String t = xxi[j];
      xxi[j] = xxi[j+1];
      xxi[j+1] = t;
    }
  }
}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

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());
```

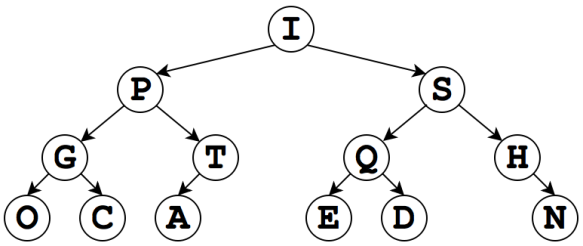
Question 29.

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?

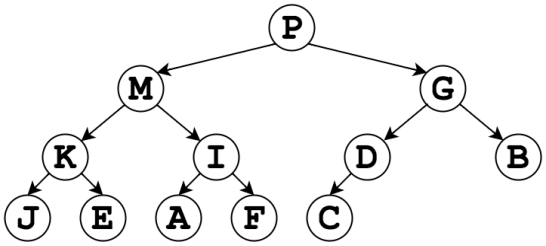
A) Min-heap

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

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

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
```

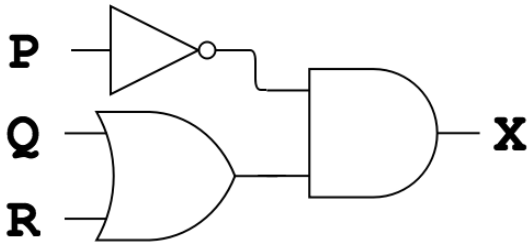
Client Code

```
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 ★

Name_____ School_____

UIL COMPUTER SCIENCE WRITTEN TEST – 2016 INVITATIONAL B

Questions (+6 points for each correct answer, -2 points for each incorrect answer)

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

FOR ADMINISTRATIVE USE ONLY

# Right:	×	6 pts	=	
# Wrong:	×	-2 pts	=	
# Skipped:	×	0 pts	=	0

	Score	Initials
Judge #1:		
Judge #2:		
Judge #3:		

★ANSWER KEY – CONFIDENTIAL★

UIL COMPUTER SCIENCE WRITTEN TEST – 2016 INVITATIONAL B

Questions (+6 points for each correct answer, -2 points for each incorrect answer)

1) D	11) E	21) A	31) C
2) A	12) C	22) C	32) E
3) C	13) C	23) B	33) D
4) A	14) A	24) A	34) B
5) B	15) D	25) D	35) C
6) E	16) E	26) B	36) A
7) B	17) D	27) E	37) D
8) B	18) D	28) C	38) E
9) D	19) B	29) A	*39) $\overline{P}$ * (Q+R)
10) A	20) C	30) D	40) – 42

** See "Explanation" section below for alternate, acceptable answers.*

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.

Explanation

1) D 42_{10} * 3_{10} = 1111110_2 = 1332_4 = 176_8 = 126_{10} = $7e_{16}$

2) A `3 * 2.5 = 7.5`

3) C `"d%d.o%o.x%x"`: The substrings,`"%d"`, `"%o"`, and `"%x"`, insert the 3 integer parameters into the output formatted as *decimal*, *octal*, and *hexadecimal* integers, respectively.

4) A `dna.indexOf("CAA")` returns a value of 1; `dna.indexOf("CA", 1)` returns a value of 1

5) B

p	q	r	!(p || !(q && !r))
0	0	0	**0**
0	0	1	**0**
0	1	0	**1**
0	1	1	**0**
1	0	0	**0**
1	0	1	**0**
1	1	0	**0**
1	1	1	**0**

6) E "Incompatible types" error occurs at compile time. `Math.round()` returns a `long`, which cannot be assigned to `int` without loss of precision.

★ANSWER KEY – CONFIDENTIAL★

7) B In addition to assigning a value of 5 to `val`, the compound assignment statement `val %= 20` also evaluates to the `int` value assigned to `val` (i.e., 5).

8) B 28 is a multiple of 4. The first conditional evaluates to `true`. All other conditions are unchecked.

9) D Loop iterates through 6 times when `control` is 64, 32, 16, 8, 4, and 2. Each iteration prints a "+" if `control` is still even *after* being halved. The 6th iteration (when `control` is initially 2 and gets halved to 1) does not print anything.

10) A Each index position from 1 through `digits.length - 1` is assigned 1 more than its predecessor.

11) E `Sum` requires that input be read as `int` values using `nextInt()`, so loop should be conditioned on the Boolean method `hasNextInt()`.

12) C Sum of all integers in file = `11 + 9 + 70 + 3 + -50 + 19 + 12 + 5 + 7 + 1 + 4 + -3` = 88

13) C `(5.0 + ((7 / 2) * 3)) = (5.0 + (3 * 3)) = (5.0 + 9) = 14.0`. Note that `7 / 2` uses integer division that truncates result to 3.

14) A As a 64-bit, floating-point value the `double` data type can store a much wider range of values than can be stored in a 32-bit `float`.

15) D Individual letters are inserted into the `List` in reverse order. Removes every `'b'` that does not immediately follow a `'b'` that was previously removed (i.e., for each pair of adjacent `'b'` characters, only the first is removed from the list). Output `String` consists of the remaining letters in the same reversed order in which they are stored in the `List`.

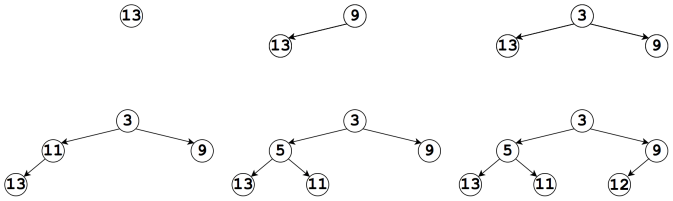
16) E
`\\D+` Equates to "1 or more non-digits".
`[int f]+` Equates to "1 or more of the 5 characters: `i`, `n`, `t`, space, or `f`".
`\\D\\D\\W?` Equates to "2 non-digits followed by 0 or 1 non-word (non-alphanumeric) characters".
`(ft )|(in)` Equates to "the substring `'ft '` (including the space) OR the substring `'in'` (no space)".

17) D While the *IEEE 754: Standard for Binary Floating-Point Arithmetic* defines distinct binary representations for negative zero and positive zero, the specification states, "*Comparisons shall ignore the sign of zero (so +0 = – 0).*" Thus, negative zero (`-1.0`)and positive zero (`0.0`) are considered equal values.

18) D The `nonNice()` method calculates the arithmetic mean of the array contents. Integer division truncates the mean before the value is returned.

19) B sum = (0 + 2 + 4 + 6 + 8) + (1+ 3 + 5 + 7 + 9) + (2 + 4 + 6 + 8) + (3 + 5 + 7+ 9) + (4 + 6 + 8) = 107

20) C The `PriorityQueue` class is implemented as a min-heap with its `toString()` method producing a level-by-level traversal of the resulting tree. The structure of the min-heap after each element is added is shown below:



21) A Bubble Sort arranges `String` values in descending order (i.e., lowercase letters > uppercase letters > digits).

22) C Depth-First Search produces a pre-order traversal of a binary tree.

23) B Breadth-First Search produces a level-by-level traversal of a binary tree.

24) A `10000111 & 00001111 = 00000111 = 7`

25) D The superclass, `Alpha`, has only 1 constructor and it requires a `String` parameter.

26) B Private `inty` field in the `Omega` class is static.

27) E All references in `Omega` to `inty` refer to the local copy of `inty` declared in the `Omega` class.

28) C `Alpha` and `Inty` are abstract and cannot be directly instantiated.

29) A The tree and `huff()` method function similarly to a Huffman code. The `int` parameter serves as a bit stream (starting with the least significant bit) containing a Huffman-encoded message. $2658_{10} = 101001100010_2$. Reading right to left, `010 = A`, `001 = C`, `100 = E`, `101 = D`.

30) D When `c == 0`, the value of node r is returned. When `r == null` (i.e., `r` references an empty tree), a `nullPointerException` is thrown. When `c != 0` and `r` is a leaf, the method results in infinite recursion.

★ANSWER KEY – CONFIDENTIAL★

31) C When `c != 0`, the method either traverses down to a child of `n`, reducing the value of `c` (i.e., approaches the base case where `c == 0`), or it restarts the traversal back at `r`. When `r == null` (i.e., `r` references an empty tree), a `nullPointerException` is thrown.

32) E `('T' = 84) < ('e' = 101) < ('r' = 114)`

33) D `4 + (((8 + 7) * 2) * (3 – 1)) = 64`

34) B Tree is complete. Every interior node is greater than its children.

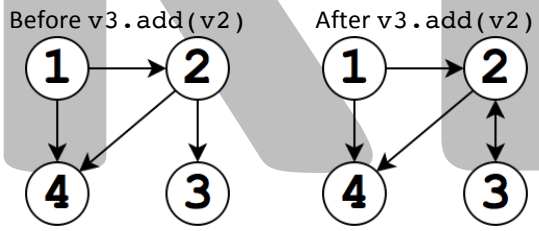
35) C

X	Y	Z	$(X + \overline{Y})(X + Z)$	$X\overline{Y} + XZ$	$\overline{X}Y + \overline{X}\,\overline{Z}$	$X + \overline{Y}Z$	$X(\overline{Y} + Z)$	$X + (\overline{Y} + Z)$
0	0	0	**0**	0	1	**0**	0	1
0	0	1	**1**	0	0	**1**	0	1
0	1	0	**0**	0	1	**0**	0	0
0	1	1	**0**	0	1	**0**	0	1
1	0	0	**1**	1	0	**1**	0	1
1	0	1	**1**	1	0	**1**	1	1
1	1	0	**1**	0	0	**1**	0	1
1	1	1	**1**	1	0	**1**	0	1

36) A Each `Struct` object represents a node in a directed graph. The `Struct` class statically stores a collection of all nodes in the graph.

37) D The `Struct` class is backed by a `TreeSet`. The `contains()` method for a `TreeSet` has $O(\log_2 N)$ performance.

38) E



39) Any answer that equivalently expresses "(Logical-NOT P) Logical-AND (Q Logical-OR R)" is acceptable (e.g., "!P && (Q || R)", "P' (Q + R)", "$\overline{P}$ (Q + R)").

40) $11010110_2 = -42_{10}$. $00101010_2 = +42_{10}$.

UIL COMPUTER SCIENCE WRITTEN TEST

2016 DISTRICT 1

MARCH 21-26, 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.
  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)
```

UIL COMPUTER SCIENCE WRITTEN TEST – 2016 DISTRICT 1

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 $100101_2 + 1111_2$?

A) 101010_2 **B)** 65_8 **C)** 104_{10} **D)** 33_{16} **E)** $1G_{36}$

Question 2.

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

A) 0 **B)** 3 **C)** 3.125 **D)** 4 **E)** 4.125

```
int x = 25;
int y = 8;
out.println((x + y) / y);
```

Question 3.

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

A) 8573764954 **B)** 61LIU **C)** UIL16

D)
```
54
49
76
73
85
```

E)
```
U
I
L
1
6
```

```
char[] uil = {'U', 'I', 'L', '1', '6'};
for (int i = uil.length - 1; i >= 0; i--)
  out.print(uil[i]);
```

Question 4.

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

A) ransmog **B)** ansmogri **C)** ansmog

D) ransmo **E)** ansmogr

```
String calvin = "Transmogrifier";
out.println(calvin.substring(2, 8));
```

Question 5.

Which of the following Boolean expressions is equivalent to the truth table for output X, as shown to the right?

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

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

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

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

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

P	Q	R	**X**
0	0	0	**0**
0	0	1	**0**
0	1	0	**1**
0	1	1	**1**
1	0	0	**0**
1	0	1	**0**
1	1	0	**0**
1	1	1	**1**

Question 6.

Which of the following outputs can never be produced by the code segment to the right?

A) 16 **B)** 17 **C)** 20 **D)** 35 **E)** 36

```
int ran = 20;
int dumb = 16;
int num = (int)(Math.random() * ran + dumb);
out.println(num);
```

Question 7.

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

A) –3.0 **B)** –3 **C)** 3 **D)** 3.3

E) No output due to an error.

```
int huey = -10;
int dewey = huey * 3;
int louie = dewey - 3;
out.println(louie %= huey);
```

Question 8.

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

```
int test = 5;
```

A) 5 **B)** five.5 **C)** five.6

D) six.eight.9 **E)** five.six.eight.9

```
switch(test) {
  case 5:  out.print("five.");
           test++;
           break;
  case 6:  out.print("six.");
           test += 3;
  case 8:  out.print("eight.");
}
out.println(test);
```

Question 9.

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

A) @@@ **B)** @@@@@

C) @@@@@@ **D)** @@@@@@@

E) The code segment prints an infinite string of @ characters.

```
int at = 3;
do {
  out.print("@");
  at *= 3;
} while (at < 2500);
```

Question 10.

What is the return value of the following invocation of the `process()` method from a client class?

```
int[] auto = {13, 9, 3, 11, 5, 12};
out.println( process(auto) );
```

A) 0 **B)** 17 **C)** 33 **D)** 40 **E)** 53

```
public static int process(int[] a) {
  int b = 0;
  int c = 0;
  for (int d : a) {
    if (d < b) c += d;
    b = d;
  }
  return c;
}
```

Question 11.

What is the output of the code segment to the right if the user enters the following line of console input?

```
28 11 30 15 33 45 END
```

A) 13 11 8 15 3 0 **B)** 15 4 4 4 4 4

C) 15 6 6 6 6 6 **D)** 13 11 0 0 3 0

E) 13 11 8 7 5 0

```
Scanner user = new Scanner(System.in);
int alfa = 15;
while (user.hasNextInt()) {
  int bravo = user.nextInt();
  alfa = bravo % alfa;
  out.print(alfa + " ");
}
```

Question 12.

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

A) 994 **B)** 1000 **C)** 1994 **D)** 2000 **E)** 3992

```
int half = 0;
for (int i = 1000; i > 0; i /= 2)
   half += i;
out.println(half);
```

Question 13.

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

A) 0 **B)** 5 **C)** 23 **D)** 55 **E)** 1472

```
out.println(23 | 6 << 3 ^ 7 & 13);
```

Question 14.

Which of the following abstract data types would be the most optimal choice for implementing the database for a mobile app that allows a customer to look up the price of a product by scanning its barcode (UPC)?

A) LinkedList **B)** Stack **C)** HashMap **D)** TreeSet **E)** PriorityQueue

Question 15.

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

A) [0, 1, 3, 4, 24, 30, 36, 42, 48, 54]

B) [0, 6, 12, 6, 24, 30, 12, 42, 48, 18]

C) [0, 1, 1, 2, 3, 30, 36, 42, 48, 54]

D) [0, 1, 12, 2, 3, 30, 36, 42, 48, 54]

E) [0, 6, 12, 18, 24, 30, 36, 42, 48, 54]

```
List<Byte> bytes = new ArrayList<Byte>();
for (int i = 0; i < 10; i++) {
  bytes.add((byte)(i * 6));
  if (i % 3 == 0)
    bytes.set((byte)(i / 2), (byte)(i / 3));
}
out.println(bytes);
```

Question 16.

What is printed by the following invocation of the `findX()` method from a client class?

```
out.println( findX(9) );
```

A) 04 **B)** 34 **C)** 09 **D)** 39 **E)** 123-1

Question 17.

What is printed by the following invocation of the `findX()` method from a client class?

```
out.println( findX(6) );
```

A) 1310 **B)** 123-1 **C)** 2310 **D)** 210 **E)** 010

```
public static int findX(int x) {
  int i = -1;
  int code = 0;
  int[] a = {7, 1, 3, 4, 9, 8, 2, 5, 0, 9};
  try {
    while (a[++i] != x) {}
  } catch (RuntimeException e) {
    code = code * 10 + 1;
  } catch (Exception e) {
    code = code * 10 + 2;
  } finally {
    code = code * 10 + 3;
  }
  out.print(code);
  return i;
}
```

Question 18.

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

A) This **B)** That **C)** the Other

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

```
String[] label = {"This", "That",
                  "the Other" };
if (label[0].compareTo(label[1]) < 0)
  if (label[0].compareTo(label[2]) > 0)
    out.println(label[0]);
  else
    out.println(label[2]);
else if (label[2].compareTo(label[1]) < 0)
    out.println(label[1]);
else
    out.println("null");
```

Question 19.

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

A) 10 **B)** 14 **C)** 19 **D)** 24

E) No output due to an error.

```
out.println(Integer.toString(19, 15));
```

Question 20.

Which of the following strings does NOT match the regular expression shown to the right?

A) aaabbbaba **B)** aabbaabba **C)** abba

D) abbbabbba **E)** abaabbaaba

a+bb(a*b)*a

Question 21.

What is printed by the following invocation of the `mangle()` method from a client class?

```
out.println(mangle("DistrictUIL"));
```

A) iDistrctUIL **B)** isiDrtUtcLI

C) DitrsctILUi **D)** iULItcsrtiD

E) ILctUtrDisi

```
public static String mangle(String s) {
  if (s.length() == 0) return "";
  int i = s.length() / 2;
  char c = s.charAt(i);
  String s1 = s.substring(0, i);
  String s2 = s.substring(i+1);
  return c + mangle(s1) + mangle(s2);
}
```

Question 22.

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

A)
```
42.43.
63.64.65.
84.85.86.87.
```

B)
```
7.
9.10.11.
11.12.13.14.
```

C)
```
43.
63.64.65.
83.84.85.86.
```

D)
```
6.7.
9.10.11.
12.13.14.15.
```

E) 7.9.10.11.11.12.13.14.

```
for (int six = 4; six < 10; six += 2) {
  for (int two = six / 2; two < six; two++)
    out.print(six + two + ".");
  out.println();
}
```

Question 23.

Which of the following algorithms is implemented by the `alg()` method to the right?

A) Selection Sort **B)** Binary Search

C) Merge Sort **D)** Insertion Sort

E) Quicksort

Question 24.

What is the expected runtime performance for the `alg()` method in the best case? Choose the most restrictive answer.

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

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

Question 25.

What is the output of line **<#1>** in the **Client Code** to the right?

A) –1 **B)** 0 **C)** 7 **D)** 8 **E)** 28

Question 26.

What is the output of line **<#2>** in the **Client Code** to the right?

A) `District`

B) `[D, c, i, i, r, s, t, t]`

C) `ttsriicD`

D) `[D, i, s, t, r, i, c, t]`

E) `[t, t, s, r, i, i, c, D]`

```
static int alg(char[] a) {
  int n = -1;
  for (int i = 0; i < a.length; i++) {
    int m = i;
    for (int j = a.length-1; j > i; j--) {
      if (a[m] < a[j])
        m = j;
    }
    help(a, i, m);
    n++;
  }
  return n;
}

static void help(char[] a, int x, int y){
  char z = a[x];
  a[x] = a[y];
  a[y] = z;
}
```

Client Code

```
String dist = "District";
char[] data = dist.toCharArray();
out.println(alg(data));      //<#1>
String s = Arrays.toString(data);
out.println(s);              //<#2>
```

Question 27.

Given the code segment to the right, how many of the elements of the `chunks` array are empty strings?

A) 0 **B)** 1 **C)** 2 **D)** 3 **E)** 5

```
String block = "to be or not to be";
String axe = ".\\s.";
String[] chunks = block.split(axe);
```

Question 28.

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

A) `bceg`

B) `bdddff`

C) `abcdefg`

D) `aabbcddddefffg`

E) No output due to an error.

```
Queue<Object> que = new LinkedList<>();
String pattern = "aabbcddddefffg";
for (char c : pattern.toCharArray())
  que.add(c);
while (!que.isEmpty()) {
  if (!que.remove().equals(que.peek()))
    out.print(que.remove());
}
```

Question 29.

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

A) 0 **B)** 5 **C)** 10 **D)** 84 **E)** 3125

```
out.println(2 + 3 ^ 4 + 1);
```

Question 30.

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

A) `[1, 5, 8, 10]`

B) `[5, 6, 9]`

C) `[1, 2, 3, 4]`

D) `[2, 6, 7]`

E) No output due to an error.

```
int[][] grid = { {1, 2, 3, 4}, {5, 6, 7},
                 {8, 9}, {10} };
for (int i = 0; i < grid.length; i++)
  for (int j = 0; j < grid[i].length; j++)
    grid[i][j] = grid[j][i];
out.println(Arrays.toString(grid[1]));
```

Question 31.

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

```
Customer moe = new Cheapo();
moe.pay(5.00);
moe.pay(2.00);
moe.pay(10.50);
out.println(moe);
```

A) `$17.5 + $0.0`

B) `$17.5 + $3.5`

C) `$17.5 + $8.75`

D) `$10.5 + $0.0`

E) No output due to an error.

Question 32.

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

```
Customer larry = new MoneyBags();
larry.pay(5.00);
larry.pay(2.00);
larry.pay(10.50);
out.println(larry);
```

A) `$17.5 + $0.0 (0%)`

B) `$17.5 + $0.0 (50%)`

C) `$17.5 + $8.75 (50%)`

D) `$17.5 + $3.5 (20%)`

E) No output due to an error.

Question 33.

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

```
Customer curly = new Customer();
curly.pay(5.00);
curly.pay(2.00);
curly.pay(10.50);
out.println(curly);
```

A) `$17.5 + $0.0`

B) `$17.5 + $3.5`

C) `$17.5 + $8.75`

D) `$10.5 + $0.0`

E) No output due to an error.

```
public interface Customer {
  public double rate = 1.20;
  public double pay(double amount);
}

public class Cheapo implements Customer {
  public double amounts;
  public double tips;
  public double rate = 0.00;

  public Cheapo() {}

  public double pay(double amount) {
    amounts += amount;
    double tip = amount * rate;
    tips += tip;
    return amount + tip;
  }

  public String toString() {
    return "$" + amounts + " + $" + tips;
  }
}

public class MoneyBags extends Cheapo {
  public double rate = 1.50;

  public double pay(double amount) {
    return super.pay(amount);
  }

  private String getRate() {
    return (int)((rate - 1) * 100) + "%";
  }

  public String toString() {
    return super.toString() + " (" +
           getRate() + ")";
  }
}
```

Question 34.

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

A) `five [one, two]`

B) `[two, four] two`

C) `five two null`

D) `four two null`

E) `five two`

```
Map<String, String> map = new HashMap<>();
map.put("one", "two");
map.put("two", "two");
map.put("four", "five");
map.put("five", "one");
map.put("one", "four");
out.println(map.get("one") + " " +
            map.get("two") + " " +
            map.get("three"));
```

Question 35.

Which of the following is a post-order traversal of the binary tree to the right?

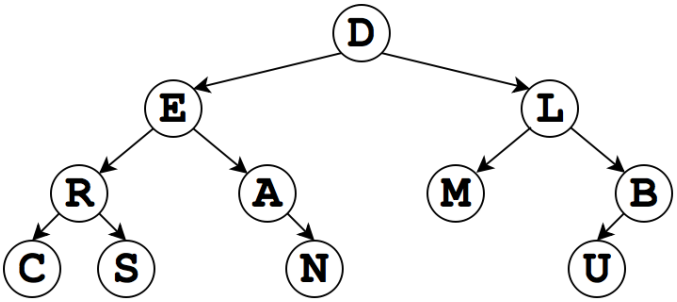
A) CRSEANDMLUB

B) DELRAMBCSNU

C) CSRNAEMUBLD

D) DERCSANLMBU

E) CRSEANMLUBD



Question 36.

Given the truth table to the right with inputs P, Q, and R, which of the following is a valid statement about output X?

A) X will always be 0 whenever the values of P and R are different from each other.

B) X will always be 1 whenever P is 1.

C) X will always be 1 whenever exactly 2 inputs are both 1.

D) X will always be 0 whenever Q is 0.

E) More than one of these statements is valid.

P	Q	R	X
0	0	0	**0**
0	0	1	**0**
0	1	0	**0**
0	1	1	**1**
1	0	0	**0**
1	0	1	**1**
1	1	0	**1**
1	1	1	**1**

Question 37.

What is the 8-bit, 2's complement representation of -100_{10}?

A) 10011011_2 **B)** -00000100_2 **C)** 11100100_2 **D)** 10011100_2 **E)** 10000100_2

Question 38.

Which of the following set of inputs for the logic diagram to the right will result in a `false` outputs for both X and Y?

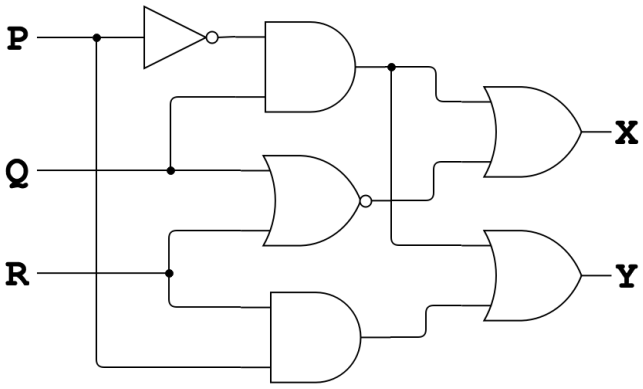
A) `P = false; Q = false; R = false`

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

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

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

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



Question 39.

Write a simplified, Boolean expression that is equivalent to the expression to the right. Your answer should include as few logical operators as possible.

Write your answer on the answer sheet.

```
(A * B) + (C * B)
```

Question 40.

What is the postfix notation (reverse Polish notation) for the arithmetic expression to the right?

Write your answer on the answer sheet.

```
7 - (3 * 4)
```

★ DOUBLE-CHECK YOUR ANSWERS ★

★ANSWER KEY – CONFIDENTIAL★

UIL COMPUTER SCIENCE WRITTEN TEST – 2016 DISTRICT 1

Questions (+6 points for each correct answer, -2 points for each incorrect answer)

1) E	11) E	21) B	31) A
2) D	12) C	22) D	32) B
3) B	13) D	23) A	33) E
4) C	14) C	24) B	34) D
5) A	15) D	25) C	35) C
6) E	16) B	26) E	36) C
7) B	17) A	27) D	37) D
8) C	18) D	28) A	38) B
9) D	19) B	29) A	*39) B * (A + C)
10) B	20) E	30) B	*40) 734*–

** See "Explanation" section below for alternate, acceptable answers.*

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.

Explanation

1) E $100101_2 + 1111_2 = 110100_2 = 64_8 = 52_{10} = 34_{16} = 1G_{36}$

2) D `(25 + 8) / 8 = 33 / 8 = 4` (integer division)

3) B Loop iterates backward through array, printing Unicode characters in reverse order all on 1 line.

4) C `substring(int begin, int end)`: Returns the substring from index `begin` through index `(end - 1)`.

5) A

P	Q	R	**X**	A)	B)	C)	D)	E)
0	0	0	**0**	0	1	0	0	0
0	0	1	**0**	0	1	0	1	0
0	1	0	**1**	1	1	1	1	0
0	1	1	**1**	1	1	0	1	0
1	0	0	**0**	0	0	1	0	0
1	0	1	**0**	0	1	0	1	0
1	1	0	**0**	0	1	1	0	0
1	1	1	**1**	1	1	0	1	1

6) E The code segment can produce outputs in the range of 16 through 35, inclusive.

7) B `huey = -10; dewey = -30; louie = -33; -33 % -10 = -3`

★ANSWER KEY – CONFIDENTIAL★

8) C The `switch` matches on the `case` where `test = 5`, prints "`five`", increments `test` to 6, then breaks out of the `switch-case` statement before printing the final value of `test` (6).

9) D Prints an "`@`" when `at = 3`, 9, 27, 81, 243, 729, and 2187. Exits the loop when `at = 6561`.

10) B `9 + 3 + 5 = 17`

11) E Value of `alfa`, `bravo`, and output at the point of the `print()` invocation in each iteration of the loop:

```
 alfa: 15 13 11  8  7  5
bravo: 28 11 30 15 33 45
```
Output: 13 11 8 7 5 0

12) C `half = 0 + 1000 + 500 + 250 + 125 + 62 + 31 + 15 + 7 + 3 + 1 = 1994`

13) D
```
= (23 | ((6 << 3) ^ (7 & 13)))
= (23 | (48 ^ (7 & 13))
= (23 | (48 ^ 5))
= (23 | 53)
= 55
```

14) C The barcode (UPC) serves as the lookup *key* and the product price is the *value* associated with that key. O(1) to add/update product prices. O(1) to look up the price of a product.

15) D
```
bytes = []
bytes = [0]          (when i = 0, set bytes[0/2] = 0/3)
bytes = [0, 6]
bytes = [0, 6, 12]
bytes = [0, 1, 12, 18]          (when i = 3, set bytes[3/2] = 3/3)
bytes = [0, 1, 12, 18, 24]
bytes = [0, 1, 12, 18, 24, 30]
bytes = [0, 1, 12, 2, 24, 30, 36]          (when i = 6, set bytes[6/2] = 6/3)
bytes = [0, 1, 12, 2, 24, 30, 36, 42]
bytes = [0, 1, 12, 2, 24, 30, 36, 42, 48]
bytes = [0, 1, 12, 2, 3, 30, 36, 42, 48, 54]          (when i = 9, set bytes[9/2] = 9/3)
```

16) B Return value, `i`, increments to index of first occurrence of `x` (`i = 4`). No exceptions are thrown and the `finally` clause is always executed. The method prints "`3`" before returning and the client class prints "`4`".

17) A Return value, `i`, increments beyond the end of the array causing an `ArrayIndexOutOfBoundsException` to be thrown when `i = 10`. The exception is caught by the first `catch()` clause (`code = 1`) and the `finally` clause is always executed (`code = 13`). Note that `ArrayIndexOutOfBoundsException extends IndexOutOfBoundsException extends RuntimeException extends Exception`. The method prints "`13`" before returning and the client class prints "`10`".

18) D "`That`" < "`This`" < "`the Other`" when compared lexicographically (case-sensitive).

19) B $19_{10} = 14_{15}$

20) E The regular expression requires 2 `b`'s following 1 or more leading `a`'s.

21) B Recursively produces a pre-order traversal of the tree whose in-order traversal is the parameter `String s`.

22) D Outer loop iterates `six` through values of 4, 6, and 8. Inner loop iterates `two` through values of 2 through 3, 3 through 5, and 4 through 7, respectively for each pass through the outer loop.

23) A Sorts the array into descending order using selection sort.

24) B Selection sort yields $O(N^2)$ performance in the best, average, and worst cases.

25) C Selection sort performs `N` swaps, incrementing the return value from `-1` through `N-1` (i.e., 7), including cases where an item is swapped with itself (i.e., `i == m`).

26) E `data` is a `char[]` sorted in descending order by Unicode value.

27) D `chunks = ["t", "", "", "o", "", "e"]`. The regular expression matches on "`o b`", "`e o`", "`r n`", "`t t`", and "`o b`".

28) A `remove()` causes an item to be removed from the queue, but `peek()` does not remove the item from the queue.

★ANSWER KEY – CONFIDENTIAL★

29) A `((2 + 3) ^ (4 + 1)) = 5 XOR 5 = 0`

30) B `grid = [[1, 5, 8, 10], [5, 6, 9], [8, 9], [10]]`

31) A `amounts = 5.00 + 2.00 + 10.50 = 17.50; tips = 0.00 + 0.00 + 0.00 = 0.00`

32) B The `pay()` method defined in the `Cheapo` class uses the `private rate` field declared in the `Cheapo` class (`0.00`) and does not use either the `private rate` field declared in the `MoneyBags` subclass (`1.50`) or the `private rate` field declared in the `Customer` interface (`1.20`). But the `getRate()` method in the `MoneyBags` subclass uses the `private rate` field declared in the `MoneyBags` subclass (`1.50`).

33) E `Customer` is an `interface`. It cannot be directly instantiated with `new Customer()`.

34) D

Keys	Values
one	~~two~~ four
two	two
four	five
five	one

35) C Pre-order: DERCSANLMBU. In-Order: CRSEANDMLUB. Level-by-level: DELRAMBCSNU

36) C `X = (Q * R) + (P * R) + (P * Q)`. If any 2 inputs are 1 (i.e., Q and R, P and R, or P and Q), the output is 1. Note that the correct answer choice only addresses cases in which "exactly 2" inputs are 1 and makes no statement about the output if all 3 inputs are true (i.e., the output could be either 0 or 1 in that case).

37) D $100_{10} = 01100100_2$; $-100_{10} = 10011100_2$; 1's complement of $-100_{10} = 10011011_2$

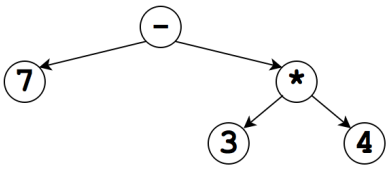
38) B

P	Q	R	**X**	**Y**
0	0	0	**1**	**0**
0	0	1	**0**	**0**
0	1	0	**1**	**1**
0	1	1	**1**	**1**
1	0	0	**1**	**0**
1	0	1	**0**	**1**
1	1	0	**0**	**0**
1	1	1	**0**	**1**

39) Any answer that equivalently expresses "(NOT B) Logical-AND (A Logical-OR C)" is acceptable (use of parentheses for correctly enforcing order of operations is required):

```
B(A + C)          B(C + A)          (A + C)B          (C + A)B
B * (A + C)       B * (C + A)       (A + C) * B       (C + A) * B
B && (A || C)     B && (C || A)     (A || C) && B     (C || A) && B
B and (A or C)    B and (C or A)    (A or C) and B    (C or A) and B
```

40)



Postfix (reverse Polish) notation: `734*–`
Prefix (Polish) notaton: `–7*34`
Infix notation: `7–(3*4)`

Conference _____ Code Number _____

UIL COMPUTER SCIENCE WRITTEN TEST

Questions (+6 points for each correct answer, -2 points for each incorrect answer)

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

FOR ADMINISTRATIVE USE ONLY

# Right:	×	6 pts	=	
# Wrong:	×	-2 pts	=	
# Skipped:	×	0 pts	=	0

	Score	Initials
Judge #1:		
Judge #2:		
Judge #3:		

Computer Science Competition
District 1 2016
Programming Problem Set

I. General Notes

1. Do the problems in any order you like. They do not have to be done in order from 1 to 12.

2. All problems have a value of 60 points.

3. There is no extraneous input. All input is exactly as specified in the problem. Unless specified by the problem, integer inputs will not have leading zeros. Unless otherwise specified, your program should read to the end of file.

4. Your program should not print extraneous output. Follow the form exactly as given in the problem.

5. A penalty of 5 points will be assessed each time that an incorrect solution is submitted. This penalty will only be assessed if a solution is ultimately judged as correct.

II. Names of Problems

Number	Name
Problem 1	Madison
Problem 2	Nicholas
Problem 3	Oksana
Problem 4	Patricio
Problem 5	Rishabh
Problem 6	Susan
Problem 7	Tomas
Problem 8	Violeta
Problem 9	Walter
Problem 10	Ximena
Problem 11	Yash
Problem 12	Zhenya

1. Madison

Program Name: Madison.java **Input File: madison.dat**

Madison always tells the truth, well, sometimes. She uses the words "and", "or", "not", and "xor" when talking about two things at the same time. For example she would say, "I took out the trash and fed the dog". If indeed she did both things, then she would be telling the truth. However, if she really did not feed the dog, then she would not be telling the truth.

Now if she said, "I took out the trash or I fed the dog.", she would be telling the truth overall if she did indeed feed the dog, but did not take out the trash. As long as at least one of those two situations was true, she would be telling the truth.

What's really confusing is when she says, "xor". This means that she is telling the truth when only one of the two things is true. For example, if she says, "I fed the dog xor I cleaned my room.", she would only be telling the truth if she only did one of those two things. If she did both, it would be a lie, or if she did neither, that would also be false.

We'll represent two or three situations using the letters A, B and C, in some kind of combination expression using NOT(!), AND(*), XOR(^) and OR(+), in that order of priority. In other words, NOT has top priority, and OR has lowest priority. You'll also know the true or false state of each letter.

For example, for the expression A+B, and the values 11, which means A is 1 and B is 1, the final result will be true, because true OR true is true.

For the expression A*B, and the values 10, the final result would be false, since true AND false is false.

There may be parentheses involved as well, in which case the expression inside would be evaluated first. There will be no instances of nested parentheses. It is possible for any of the variables to be used more than once in an expression, like A*B+A*C.

Input: Several lines of data, each line with an expression containing no spaces, followed by a two or three digit string of zeros and ones, representing the corresponding values of the variables in the expression.

Output: The final true or false value of the expression, given the actual values of each variable.

Sample Input:

```
A+B 11
A*B 10
A+B*C 101
A^B 01
!(A+B) 10
```

Sample Output:

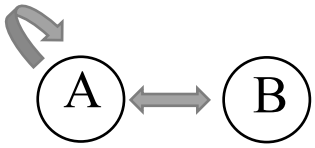
```
true
false
true
true
false
```

2. Nicholas

Program Name: Nicholas.java **Input File: nicholas.dat**

Nicholas is learning about how to manually count path lengths in a graph, such as the one shown here, using adjacency matrices, but still needs your help to confirm his counting. He started with this simple example, which produces the following adjacency matrix:

```
  A B
A 1 1
B 1 0
```



He has learned that this matrix means that given two vertices A and B in the graph above, there is a connection from A back to itself, and a two-way connection from A to B. To count the number of paths of length one, or direct connections in the graph, all he has to do is count the number of 1s in the graph, three in this case, represented in letter notation as AA, AB, and BA. AA means that the connection starts and ends at A, AB means it starts at A and ends at B, and so on.

However, counting the number of two-hop paths is a little more involved. He can see that AAA is a possibility, and then ABA, and then BAB, but wonders if AA could be combined with B. His teacher says, "Yes, it can.", so he adds AAB and BAA to his count, making a total of five 2-hop paths.

"What about 3-hop paths?", he wonders. Hmmm, let's see, starting from A there would be AAAA, AAAB, AABA, ABAA, and ABAB. Starting from B, the 3-hop paths are BAAA, BAAB, and BABA. Altogether, that would make eight 3-hop paths within this graph.

Your job is to take any graph with at least two, but no more than five vertices, and help Nicholas confirm his counting of the number of paths of any length from 1 to 3 hops.

Input: Several lines of data, each consisting of an integer N, indicating a graph with N nodes, followed by an integer P, the path length to calculate, followed by N strings, each of length N, containing 0s and 1s, indicating the adjacency matrix for the graph.

Output: The number of P length paths in the given matrix.

Sample Input:

```
2 1 11 10
2 2 11 10
2 3 11 10
3 1 111 001 001
4 2 0100 0010 0001 1000
```

Sample Output:

```
3
5
8
5
4
```

3. Oksana

Program Name: Oksana.java Input File: None

Oksana loves to play the Connect Dots game, but gets really tired of drawing out the dots on paper. She wants to be able to print them out, but needs your help.

Below is a picture of an 8X8 grid of dots. The dots in each row have a space between, but the rows do not. Write a program to output this exact pattern.

Input: None

Output: An 8X8 grid of dots as shown below, with a single space following each dot, but no blank lines between rows.

Expected Output:

```
* * * * * * * *
* * * * * * * *
* * * * * * * *
* * * * * * * *
* * * * * * * *
* * * * * * * *
* * * * * * * *
* * * * * * * *
```

4. Patricio

Program Name: Patricio.java Input File: patricio.dat

Patricio lives in a dynamic world. The grounds on which he lives are magical and will shift a little bit each day. He has built a rocket and wants to know the best time and place from which to launch it. The best place to launch his rocket will be the place of highest elevation. Luckily, Patricio has lived here for a long while, so he has derived a mathematical formula to represent how the grounds will shift in the next few days. These formulas can sometimes be rather complex; he needs your help to determine when and where to launch his rocket.

The formulas are a function of position coordinates on the surface (x, y) and the day, represented by t. These formulas that Patricio has derived can contain parentheses, exponents(^), multiplication(*), division(/), addition(+), subtraction(-) and standard order of operation applies. Also, as a common shorthand, if there are two variables together without an operation between, multiplication is assumed.

Input: The first integer will represent the number of data sets to follow. The first line will be the expression of elevation derived by Patricio. The next line will be the maximum values for x, y, and z, respectively. Single spaces will separate all operands and operators.

Output: The sentence **"Patricio should launch at x=**<*the x coordinate of the highest elevation*> **and y=**<*the y coordinate of the highest elevation*> **at t=**<*the day Patricio should launch*> **at a height of** <*the maximum elevation at which Patricio can launch*>**."** The x and y coordinate should be rounded to one decimal place, and t should be rounded to the nearest day (integer).

Assumptions: All division operations in Patricio's formulas are floating point division; there is no integer division. Division by 0 is considered invalid, and is not considered infinity. The variables x, y, and t can be zero, but they cannot be negative. All formulas Patricio derives are dependent on x, y, and t.

Sample Input:

```
3
z = x^3 + 50x + t^2 + x^2y + x^y + xyt
5.0 5.0 10
z = x^2 + y^2 + t^2
10.0 10.0 100
z = -5x^2 + 4x + -5y^2 + 4y + -5t^2 + 4t
2.0 2.0 10
```

Sample Output:

```
Patricio should launch at x=5.0 and y=5.0 at t=10 at a height of 3975.0.
Patricio should launch at x=10.0 and y=10.0 at t=100 at a height of 10200.0.
Patricio should launch at x=0.4 and y=0.4 at t=0 at a height of 1.6.
```

5. Rishabh

Program Name: Rishabh.java **Input File: rishabh.dat**

In researching the concept of string parity, Rishabh encountered some work done by Richard Hamming in 1950. His work produced an innovative way to improve this error-checking process to ensure data is accurately and reliably transmitted. Parity is often referred to as ODD or EVEN, by adding up the bit values in a string. If EVEN parity is used, the string 10110101, which has 5 bits, is appended with the value 1 to make the total number of bits even. If ODD parity is used, the same string would be appended with a 0, to maintain an odd total for all of the bits, resulting in 101101010 as the transmitted string.

Hamming's work produced not only a way to check IF there is an error, but WHERE the error is in the string. Supposedly, a parity code based on Hamming's technique is said to be a perfect code in reliably transmitting and fixing strings that become corrupted. Here is an example using even parity. For a bit string of length 8, which is 2^3, 4 bits can be added to create Hamming parity string. For a string of 2^N bits, the leftmost bit is considered position 1 and the rightmost bit is position 2^N. The parity bits will be placed at each position that is a power of 2 (i.e., at positions 1, 2, 4, 8, 16, …), with the actual data bits starting in position 3, then 5, 6, 7, 9, 10, ... and so on. For the string 10110101, which can be represented by the hex string B5, the Hammond parity string would start out as:

```
1 2 3 4 5 6 7 8 9 10 11 12
_ _ 1 _ 0 1 1 _ 0  1  0  1
```

To find the bit value in position 1, add up all of the odd position bits, with any blank counting as a zero. This would be 0+1+0+1+0+0, for a total of 2, which is even, therefore no bit needs to be added, making the value of position 1 zero.

```
1 2 3 4 5 6 7 8 9 10 11 12
0 _ 1 _ 0 1 1 _ 0  1  0  1
```

To find the bit value in position 2, start at position 2 and use two bits, then skip 2, use 2, skip 2, and so on resulting in summing the values in positions 2, 3, 6, 7, 10, and 11, for a sum of 0+1+1+1+1+0, or 4, again even parity resulting in zero for position 2.

```
1 2 3 4 5 6 7 8 9 10 11 12
0 0 1 _ 0 1 1 _ 0  1  0  1
```

To find the bit value in position 4, start at position 4 and use 4 bits, then skip 4, use 4, skip 4, and so on resulting in summing the values in positions 4,5,6,7, and 12, for a sum of 0+0+1+1+1, or 3, needing a 1 in position 4 to make it even parity. The value of the 8 bit would use the same pattern as the others: use 8, skip 8, and so on, summing the bits in positions 9, 10, 11, and 12, for a total of 2, even parity and zero for position 8. In general, for a parity bit at position P, where P is a power of 2, the general pattern continues in the same manner: start at position P then use P values, skip P values, and so on…

Here is the final transmitted string. The added parity bits in positions 1, 2, 4, and 8 are: **0010**.

```
1 2 3 4 5 6 7 8 9 10 11 12
0 0 1 1 0 1 1 0 0  1  0  1
```

For odd parity, the example above would result in an odd parity bit string of **1101**.

```
1 2 3 4 5 6 7 8 9 10 11 12
1 1 1 0 0 1 1 1 0  1  0  1
```

Input: Several strings in hex form, each followed by the word EVEN or ODD, to indicate the parity to use. It is guaranteed that the hex string will produce no more than 128 bits.
Output: The resulting Hammond parity string for the given hex string.

Sample Input:

```
B5 EVEN
B5 ODD
CF EVEN
F EVEN
ABC ODD
```

Sample Output:

```
0010
1101
0100
111
11101
```

6. Susan

Program Name: Susan.java Input File: susan.dat

Susan is experimenting with simple encoding techniques to send messages. So far, she can only get single words or word fragments, so her codes don't quite make sense, but she will persist.

The encoded word or fragment is in the form of a sentence, with the last token of the sentence (enclosed in square brackets [ ]) as the regular expression code that divides up the sentence, with an attached adjacent integer indicating the position in the resulting division of the encoded word or fragment. A regular expression enclosed in [ ] means that any of the characters, including a space, are used to divide the sentence. The [AN] example below means that the sentence is split at any occurrence of either the 'A' or the 'N'. If there is a '+', that means any combination of any of those letters will be used as a single division of the sentence.

For example, in the sentence:
`ONE FLEW EAST, ONE FLEW WEST, ONE FLEW OVER THE CUCKOO'S NEST [ ]10`
the output would be `"CUCKOO'S"` since that word is in position 10 of the resulting array.

For the sentence, `ONE IF BY LAND, TWO IF BY SEA [AN]3`
the `[AN]3` divides the sentence into the phrases `"O", "E IF BY L","", "D, TWO IF BY SE "`, with the empty string created by the `"AN"` side by side in the word `"LAND"`.

For, `"METHINKS HE DOTH PROTEST TOO MUCH [O+]2"`, the `O+` divides the sentence into `"METHINKS HE D", "TH PR", "TEST T",` and `" MUCH ",` since the `"O+"` in the expression means, "one or more Os".

Input: Several sentences, each with a code token and integer at the end.

Output: The resulting word or word fragment for each input sentence and code token.

Sample Input:
```
ONE FLEW EAST, ONE FLEW WEST, ONE FLEW OVER THE CUCKOO'S NEST [ ]10
ONE IF BY LAND, TWO IF BY SEA [AN]3
METHINKS HE DOTH PROTEST TOO MUCH [O+]2
```

Sample Output:
```
CUCKOO'S
D, TWO IF BY SE
TEST T
```

7. Tomas

Program Name: Tomas.java **Input File: tomas.dat**

In doing some unusual scientific research lately, Tomas has discovered a peculiar sequence in various naturally occurring situations, where certain measurements of growth produce the consistent pattern 1, 1, 1, 3, 5, 9, 17, and so on. He has recorded the data in stages, where stages 1, 2, and 3 are all the value 1, and then stage 4 begins showing observable growth with the value 3 (sum of the previous three values), stage 5 the value 5 (1+1+3), and so on, following the same arithmetic addition sequence. He wants to be able to predict any particular stage, and needs your help. For example, he wants to know what the value would be at up to stage 50 of this unusual growth pattern.

Input: Several integer values N (1 <= N <= 50), each representing a stage of growth according to the pattern described above, each on one line.

Output: The growth size achieved at stage N in the growth pattern described above.

Sample Input:

```
1
3
5
7
9
```

Sample Output:

```
1
1
5
17
57
```

8. Violeta

Program Name: Violeta.java Input File: violeta.dat

Violeta loves palindromes. A palindrome is a word that is the same forwards and backwards. As her fascination with them has continued, she has started noticing them more and more often. She has actually started believing that palindromes are everywhere. She has recruited you to write a program to identify the longest palindrome you can find in any given string so she can see if palindromes are indeed as common as she believes them to be.

Input: The first integer will represent the number of data sets to follow. Each data set will contain one line (with no spaces) of the word Violeta wants you to search.

Output: Each data set should have an output that is the longest palindrome found in the input string.

Assumptions: It's a big world out there, and the string Violeta gives you may not be small, so your palindrome finding program may have to be efficient to finish in a reasonable amount of time. It is guaranteed that there will always be a unique "longest length" palindrome, and that each output will be unique and be at least one character in length.

Sample Input:

```
3
abkjlkdfja123456789987654321fewrwefdsfds
beautyandthebeastisagoodfilmthequickbrownfoxjumpedoverthelazydoggodyzalehtrevodepmujxo
fnworbkciuqehtthicketofthebeast (this line is a continuation of the 2nd data set)
racardriversarereallycool
```

Sample Output:

```
123456789987654321
thequickbrownfoxjumpedoverthelazydoggodyzalehtrevodepmujxofnworbkciuqeht
racar
```

9. Walter

Program Name: Walter.java **Input File: walter.dat**

Walter works at a mail center and must decide how to mail certain items brought in by customers, according to the dimensions of the item, from small post cards to large packages. Some items don't fit any of the categories and are unmailable. If an item somehow fits into two categories, the larger one is to be used.

Below are the category descriptions that Walter can choose from. All given dimension ranges are inclusive and expressed in inches.

SMALL POST CARD: Length must be between 3.5 and 4.25, width between 3.5 and 6, and thickness between .007 and .016.
LARGE POST CARD: Length between 4.25 and 6, width between 6 and 11.5, thickness between .007 and .016.
SMALL ENVELOPE: Length - 3.5 to 6.125, width - 5 to 11.5, thickness - .016 to .25.
LARGE ENVELOPE: Length - 6.125 to 24, width - 11 to 18, thickness - .25 to .5.
SMALL PACKAGE: Item dimensions must exceed all rules for large envelope and the length plus the distance around the sides of the package other than the length equals 84 inches or less.
LARGE PACKAGE: Length plus distance around the other sides of the package exceeds 84 inches but is no more than 130 inches.

Input: Several data sets, each on one line, consisting of the three values of a mail item: length, width, thickness.

Output: The correct mail classification (in all caps) according to the specifications listed above, or the word UNMAILABLE if the item does not fit within any of the descriptions.

Sample Input:

```
4 4 .009
5 7 .013
5 7 .2
10 12 .4
20 20 40
```

Sample Output:

```
SMALL POST CARD
LARGE POST CARD
SMALL ENVELOPE
LARGE ENVELOPE
UNMAILABLE
```

10. Ximena

Program Name: Ximena.java **Input File: ximena.dat**

Like Violeta, Ximena loves palindromes, and especially likes to create her own. She'll take a sentence and make a palindrome sentence, taking each word of the sentence, creating a double palindrome for each word. A double palindrome is created when you take the first half of a word, spell it forwards, and then backwards, and then the second half of the word, spell it forwards and backwards, and then connect those two into one long word.

An even more fun thing Ximena loves to do is to take words with an odd number of characters, and do the first half of the word in forwards/backwards order, and then the second part of the word in backwards/forwards order. The middle letter of an odd length word is only used in the second half of the word.

For words of even length, she does the opposite - first half backwards/forwards, last half forwards/backwards.

For example, using her name, XIMENA, she would make a double palindrome that looks like this: MIXXIMENAANE
Using her friend's name, VIOLETA, she would create this: VIOOIVATELLETA
For her other friend, ABE, the result is AAEBBE

Input: A list of capitalized names, each at least two letters and no more than 20 in length.

Output: A double palindrome, as described and demonstrated above, each on a separate line, in all caps.

Sample Input:
```
XIMENA
VIOLETA
ABE
```

Sample Output:
```
MIXXIMENAANE
VIOOIVATELLETA
AAEBBE
```

11. Yash

Program Name: Yash.java Input File: yash.dat

Yash has just learned in computer science class about **Order of Magnitude,** or **Big O,** and wants to work out the numbers. For example, he has learned that any algorithm with an efficiency of O(1) generally takes 1 step to complete, regardless of the size of the data, with varying larger values for the other levels of efficiency, such as O(log N), O(N), O(NlogN), and O(N^2). He learned in class that for a data set of 10 items, these five values are 1, 4, 10, 40, and 100. He was bit confused by O(log N), until his teacher said, *"Think about the exponent for the power of 2 that equals or just exceeds 10."* He thought, *"OK, 2^1 is 2, 2^2 is 4, 2^3 is 8, and 2^4 is 16. That's why 4 is the log base 2, or O(logN), answer for the value 10! I get it! It's just the integer exponent of the number using base 2, that creates a value equal to or just past the number."*

He then tried to work out higher values of N, but started to get confused again, and needs your help.

Input: Several integers N, each on one line, representing the number of elements of data to be processed by an algorithm.

Output: The five values associated with five levels of Big O algorithm efficiency, as described above, for each value N, with values exceeding 999 shown using comma separation, and a single space between each value.

Sample Input:

```
10
50
100
```

Sample Output:

```
1 4 10 40 100
1 6 50 300 2,500
1 7 100 700 10,000
```

12. Zhenya

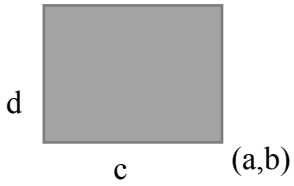
Program Name: Zhenya.java **Input File: zhenya.dat**

Zhenya works at a construction company that has been stacking clear boxes. From her perch atop the highest view on the crane, she can see the entire landscape of the box stacking. She considers her day finished when all the boxes are stacked. Because the boxes are clear and due to how high up she is though, she can only see the boxes in a 2D plane; she cannot discern the depth or order of the boxes.

This construction work is pretty atypical and boxes are considered to be stacked if they are overlapping at all, or if Zhenya can see one box through the other box. Her work day is finished if from Zhenya's vantage point, looking downward, there is at least one point on the work zone that she can see all the boxes stacked on that point. In other words, she is finished when there is a support rod that can be placed going through all the boxes vertically (through the depth of the boxes).

Please help Zhenya tell if the boxes are all stacked and she is done or if they are not.

Input: The first integer will represent the number of data sets to follow. The line will contain n groups of 4 integers, a, b, c, d. The first 2 integers (a,b) in each group of 4 represents the bottom right point of the box from Zhenya's point of view. The 3rd integer is the width (along the x axis) of the box. The 4th integer is the distance along the y-axis of the box.



Output: For each output, print "ALL STACKED" if there is a point where a rod can be placed through all of the boxes. If the boxes in their current configuration do not have this property, print "NOT STACKED".

Assumptions: Length and width will be positive.

Sample Input:

```
3
0 0 2 2 1 1 2 2 -1 -1 3 3
1 1 300 0 0 0 1 3 10 10 14 44829
0 0 10 10 2 2 5 5 4 4 1 1
```

Sample Output:

```
ALL STACKED
NOT STACKED
ALL STACKED
```

UIL COMPUTER SCIENCE WRITTEN TEST

2016 DISTRICT 2

APRIL 1-6, 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.
  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)
```

UIL COMPUTER SCIENCE WRITTEN TEST – 2016 DISTRICT

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 $110011_2 + 10111_2$?

A) 100100_2 **B)** 2202_3 **C)** 111_8 **D)** 148_{10} **E)** $4B_{16}$

Question 2.

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

A) 0 **B)** 1 **C)** 3 **D)** 4 **E)** 5

```
int x = 20;
int y = 15;
out.println(x % (x - y));
```

Question 3.

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

A) UILCS **B)** 8573767883 **C)** uilcs

D)
U
I
L
C
S

E)
85
73
76
78
83

```
int[] uil = {'U', 'I', 'L', 'C', 'S'};
for (int i : uil)
  out.println((char) i);
```

Question 4.

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

A) bral En **B)** bral Enhanc **C)** ebral En

D) bral Enh **E)** ebral Enhan

```
String hobbes = "Cerebral Enhance-o-tron";
out.println(hobbes.substring(4, 11));
```

Question 5.

Which of the following Boolean expressions is equivalent to the truth table for output X, as shown to the right?

A) (R || Q) && !P

B) (R && Q) || !P

C) !(R && Q) || P

D) R && (Q || !P)

E) !(R || Q) && P

P	Q	R	**X**
0	0	0	**0**
0	0	1	**1**
0	1	0	**0**
0	1	1	**1**
1	0	0	**0**
1	0	1	**0**
1	1	0	**0**
1	1	1	**1**

Question 6.

Which of the following outputs can never be produced by the code segment to the right?

A) 6 **B)** 7 **C)** 11 **D)** 16 **E)** 17

```
int ran = 7;
int dumb = 11;
int num = (int)(ran + dumb * Math.random());
out.println(num);
```

Question 7.

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

A) –16 **B)** –14 **C)** 14 **D)** 16

E) No output due to an error.

```
int alfa = 25;
int bravo = alfa * -3;
int charlie = bravo + 11;
out.println(charlie %= alfa);
```

Question 8.

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

```
int test = 4;
```

A) 4

B) four.seven.eight.8

C) four.7

D) four.eight.7

E) seven.four.eight.8

```
switch(test) {
  case 7:  out.print("seven.");
           test++;
  case 4:  out.print("four.");
           test += 3;
  case 8:  out.print("eight.");
}
out.println(test);
```

Question 9.

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

A) ###

B) #####

C) ######

D) #######

E) The code segment prints an infinite string of # characters.

```
int pound = 4;
do {
  out.print("#");
  pound *= pound;
} while (pound < 50000);
```

Question 10.

What is printed by the following invocation of the `crunch()` method from a client class?

```
int[] id = {109, 105, 99, 107, 101, 108};
out.println( crunch(id) );
```

A) 104 **B)** 107 **C)** 214 **D)** 314 **E)** 421

```
public static int crunch(int[] a) {
  int b = a[a.length / 2];
  int c = a[a.length / 2];
  for (int d : a) {
    if (d < b) b += d;
    if (d > c) c += d;
  }
  return (b + c) / 2;
}
```

Question 11.

What is the output of the code segment to the right if the user enters the following line of console input?

```
11 9 1 6 3 QUIT
```

A) 10 –1 2 4 –1

B) –10 2 8 –5 3

C) 10 –2 –8 5 –3

D) –10 –19 –20 –26 –29

E) –10 –8 0 –5 –2

```
Scanner parse = new Scanner(System.in);
int dog = 1;
while (parse.hasNextInt()) {
  int cat = parse.nextInt();
  out.print(dog - cat + " ");
  dog = cat;
}
```

Question 12.

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

A) 465 **B)** 945 **C)** 1905 **D)** 3840 **E)** 8415

```
int sum = 0;
for (int i = 15; i < 500; i *= 2)
   sum += i;
out.println(sum);
```

Question 13.

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

A) 0 **B)** 11 **C)** 24 **D)** 27 **E)** 31

```
out.println(28 ^ 77 >> 3 | 15 & 27);
```

Question 14.

Which of the following abstract data types would be the most optimal choice for implementing a waiting list for backordered inventory in which customers' names are stored and removed from the list in a first-come, first-served manner?

A) ArrayList **B)** Queue **C)** TreeMap **D)** HashSet **E)** Stack

Question 15.

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

A) [0, 4, 8, 12, 16, 20, 24, 28, 32, 36]

B) [0, 4, 4, 12, 8, 20, 12, 28, 16, 36]

C) [4, 8, 16, 12, 16, 20, 24, 28, 32, 36]

D) [0, 2, 8, 6, 16, 10, 24, 14, 32, 18]

E) [0, 4, 2, 12, 5, 20, 8, 28, 10, 36]

```
List<Long> longs = new ArrayList<Long>();
for (int i = 0; i < 10; i++) {
  longs.add((long) i * 4);
  if (i % 2 == 0)
    longs.set(i / 3, (long) i * 2);
}
out.println(longs);
```

Question 16.

Which of the following strings does NOT match the regular expression shown to the right?

A) 010101 **B)** 011001011 **C)** 1

D) 10010111 **E)** 0001011

```
0*1(0+1+)*
```

Question 17.

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

A) cdenull

B) feedbba

C) bdeenull

D) febb

E) No output due to an error.

```
Stack<Object> stk = new Stack<>();
String signal = "abbbccdeeeefgg";
for (char c : signal.toCharArray())
  stk.push(c);
while (!stk.isEmpty()) {
  if (stk.pop().equals(stk.pop()))
    out.print(stk.peek());
}
```

Question 18.

What is printed by the following invocation of the `shuffle()` method from a client class?

```
out.println(shuffle("Randomization"));
```

A) iinoatzdmoanR **B)** ontzaiomnRadi

C) Rnaomdztaonii **D)** aadiimnnooRtz

E) idaRnmoiaztno

```
public static String shuffle(String s) {
  if (s.length() == 0) return "";
  int i = s.length() / 2;
  char c = s.charAt(i);
  String s1 = s.substring(0, i);
  String s2 = s.substring(i+1);
  return shuffle(s1) + shuffle(s2) + c;
}
```

Question 19.

Which of the following algorithms is implemented by the `code()` method to the right?

A) Binary Search **B)** Insertion Sort

C) Selection Sort **D)** Quicksort

E) Merge Sort

Question 20.

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

A) O(N) **B)** O(N * $\log_2$ N) **C)** O($\log_2$ N)

D) O(N^2) **E)** Indeterminate

Question 21.

What is the output of line **<#1>** in the **Client Code** to the right?

A) –1 **B)** 0 **C)** 6 **D)** 12 **E)** 15

Question 22.

What is the output of line **<#2>** in the **Client Code** to the right?

A) eeimn

B) [t, e, r, , s]

C) [e, e, e, i, m]

D) ter s

E) [e, e, i, m, n]

```
static int code(char[] a, int b) {
  int n = a.length;
  if (n > 0) {
    int d = n / 2 + 1;
    if (a[d - 1] > b)
      return code(help(a, 0, d - 1), b);
    else if (a[d - 1] < b)
      return d + code(help(a, d, n), b);
    return d - 1;
  }
  return 0;
}

static char[] help(char[] a, int x, int y){
  char[] b = new char[y - x];
  for (int i = x; i < y; i++)
    b[i-x] = a[i];
  return b;
}
```

Client Code

```
String event = "computer science";
char[] data = event.toCharArray();
out.println(code(data,'e')); //<#1>

char[] sub = help(data, 5, 10);
String s = Arrays.toString(sub);
out.println(s);              //<#2>
```

Question 23.

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

A) 13 **B)** 23 **C)** 2a **D)** 31

E) No output due to an error.

```
out.println(Integer.toString(36, 13));
```

Question 24.

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

A) 0 **B)** 1 **C)** 2 **D)** 4 **E)** 7

```
String unit = "rubber baby buggy bumpers";
String knife = "b.*b";
String[] slices = unit.split(knife);
out.println(slices.length);
```

Question 25.

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

A) one fish **B)** 2 fish **C)** Red fish

D) Blue fish **E)** No output due to an error.

```
String[] label = {"one fish", "2 fish",
                  "Red fish", "Blue fish"};
if (label[0].compareTo(label[1]) > 0)
  if (label[2].compareTo(label[3]) < 0)
    out.println(label[2]);
  else
    out.println(label[3]);
else if (label[1].compareTo(label[2]) < 0)
    out.println(label[1]);
else
    out.println(label[0]);
```

Question 26.

What is printed by the following invocation of the `filter()` method from a client class?

```
out.println( filter(8) );
```

A) 0 **B)** 123 **C)** 13 **D)** 23 **E)** 3

Question 27.

What is printed by the following invocation of the `filter()` method from a client class?

```
out.println( filter(3) );
```

A) 0 **B)** 123 **C)** 13 **D)** 23 **E)** 3

```
public static int filter(int stop) {
  int i = 0;
  int code = 0;
  int[] a = {7, 1, 3, 4, 9, 8, 2, 5, 0, 9};
  try {
    while (i < stop && a[i] != 0)
      a[i] = a[i++] / a[i];
  } catch (ArithmeticException e) {
    code = code * 10 + 1;
  } catch (Exception e) {
    code = code * 10 + 2;
  } finally {
    code = code * 10 + 3;
  }
  return code;
}
```

Question 28.

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

A)
```
[.42]
[.62.63]
[.82.83.84]
```

B)
```
[.24]
[.26.36]
[.28.38.48]
```

C)
```
[.6]
[.8.9]
[.10.11.12]
```

D)
```
[]
[.26]
[.28.38]
```

E) `[.6.8.9.10.11.12]`

```
for (int one = 4; one < 10; one += 2) {
  out.print("[");
  for (int ten = 2; ten <= one / 2; ten++)
    out.print("." + ten + one);
  out.println("]");
}
```

Question 29.

Given the truth table to the right with inputs P, Q, and R, which of the following is a valid statement about output X?

A) X will always be 1 whenever R is 1.

B) X will always be 0 whenever Q is 0.

C) X will always be 0 whenever exactly 2 inputs are both 1.

D) X will always be 0 whenever the values of P and R are different from each other.

E) More than one of these statements is valid.

P	Q	R	X
0	0	0	**0**
0	0	1	**0**
0	1	0	**1**
0	1	1	**0**
1	0	0	**0**
1	0	1	**1**
1	1	0	**0**
1	1	1	**1**

Question 30.

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

A) 1 **B)** 3 **C)** 8 **D)** 10 **E)** 124

```
out.println(2 ^ 3 | 6 - 4);
```

Question 31.

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

```
Employee bart = new Slacker();
bart.paycheck(4);
bart.paycheck(2);
out.println(bart);
```

A) `$45.0 Try harder. (0.75)`

B) `$75.0 Good job! (1.25)`

C) `$75.0 Good job! (0.75)`

D) `$60.0 Thank you! (1.00)`

E) No output due to an error.

Question 32.

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

```
Employee lisa = new Worker();
lisa.paycheck(4);
lisa.paycheck(2);
out.println(lisa);
```

A) `$75.0 Good job!`

B) `$60.0 Thank you.`

C) `$25.0 Good job!`

D) `$45.0 Try harder.`

E) No output due to an error.

Question 33.

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

```
Employee maggie = new Employee();
maggie.paycheck(4);
maggie.paycheck(2);
out.println(maggie);
```

A) `$45 Try harder.`

B) `$75.0 Good job!`

C) `$60.0 Thank you.`

D) `$25.0 Good job!`

E) No output due to an error.

```
public interface Employee {
  public double bonus = 1.00;
  public double paycheck(int hours);
}

public class Worker implements Employee {
  public double bonus = 1.25;
  public double payRate = 10.00;
  public double wages;

  public double paycheck(int hours) {
    double pay = hours * payRate;
    wages += pay * bonus;
    return pay;
  }

  private String comment() {
    if (bonus < 1.0) {
      return "Try harder.";
    }
    else if (bonus > 1.0) {
      return "Good job!";
    }
    else {
      return "Thank you.";
    }
  }

  public String toString() {
    return "$" + wages + " " + comment();
  }
}

public class Slacker extends Worker {
  public double bonus = 0.75;

  public String toString() {
    return super.toString() + " (" +
           bonus + ")";
  }
}
```

Question 34.

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

A) `Doc null Bashful`

B) `Grumpy Happy Dopey`

C) `Doc null [Bashful, Dopey]`

D) `Grumpy [Dopey, Happy] null`

E) `Doc null Dopey`

```
Map<String, String> map = new TreeMap<>();
map.put("Dopey", "Doc");
map.put("Sleepy", "Sneezy");
map.put("Grumpy", "Bashful");
map.put("Happy", "Doc");
map.put("Grumpy", "Dopey");
out.println(map.get("Dopey") + " " +
            map.get("Doc") + " " +
            map.get("Grumpy"));
```

Question 35.

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

A) `[2, 6, 7]`

B) `[1, 2, 3, 4]`

C) `[5, 6, 9]`

D) `[1, 5, 8, 10]`

E) No output due to an error.

```
int[][] grid = { {1, 2, 3, 4}, {5, 6, 7},
                 {8, 9}, {10} };
for (int i = 0; i < grid.length; i++)
  for (int j = 0; j < grid[i].length; j++)
    grid[j][i] = grid[i][j];
out.println(Arrays.toString(grid[1]));
```

Question 36.

Which of the following set of inputs for the logic diagram to the right will result in a `true` outputs for both `X` and `Y`?

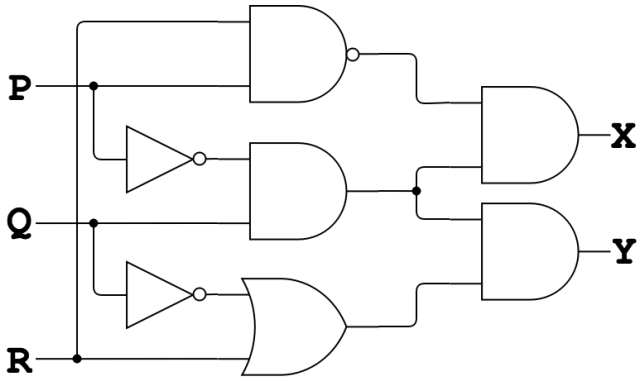
A) `P = false; Q = false; R = true`

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

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

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

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



Question 37.

Which of the following is a pre-order traversal of the binary tree to the right?

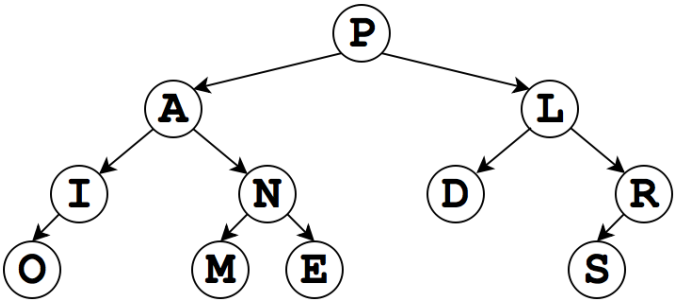
A) PAIONMELDRS

B) POIAMNEDLSR

C) PALINDROMES

D) OIAMNEPDLSR

E) OIMENADSRLP



Question 38.

What is the 8-bit, 2's complement representation of -111_{10}?

A) 10000111_2 **B)** 10010001_2 **C)** 10010000_2 **D)** -00000111_2 **E)** 01101111_2

Question 39.

What is the prefix notation (Polish notation) for the arithmetic expression to the right?

Write your answer on the answer sheet.

```
(2 + 8) / 3
```

Question 40.

Write a simplified, Boolean expression that is equivalent to the expression to the right. Your answer should include as few logical operators as possible.

Write your answer on the answer sheet.

```
(X * Y + Z) * (X + Z)
```

★ DOUBLE-CHECK YOUR ANSWERS ★

★ANSWER KEY – CONFIDENTIAL★

UIL COMPUTER SCIENCE WRITTEN TEST – 2016 DISTRICT 2

Questions (+6 points for each correct answer, -2 points for each incorrect answer)

1) B	11) B	21) D	31) C
2) A	12) B	22) B	32) A
3) D	13) E	23) C	33) E
4) A	14) B	24) C	34) E
5) D	15) C	25) D	35) A
6) A	16) B	26) C	36) C
7) B	17) D	27) E	37) A
8) D	18) C	28) B	38) B
9) A	19) A	29) D	39) /+283
10) E	20) C	30) B	*40) (X * Y) + Z

** See "Explanation" section below for alternate, acceptable answers.*

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.

Explanation

1) B $110011_2 + 10111_2 = 1001010_2 = 2202_3 = 112_8 = 74_{10} = 4A_{16}$

2) A `20 % (20-15) = 20 % 5 = 0`

3) D The loop iterates through the integer equivalents of the characters in the array, but prints each as a Unicode character, 1 character per line.

4) A `substring(int begin, int end)`: Returns the substring from index `begin` through index `(end - 1)`.

5) D

P	Q	R	**X**	A)	B)	C)	D)	E)
0	0	0	**0**	0	1	1	0	0
0	0	1	**1**	1	1	1	1	0
0	1	0	**0**	1	1	1	0	0
0	1	1	**1**	1	1	0	1	0
1	0	0	**0**	0	0	1	0	1
1	0	1	**0**	0	0	1	0	0
1	1	0	**0**	0	0	1	0	0
1	1	1	**1**	0	1	1	1	0

6) A The code segment can produce outputs in the range of 7 through 17, inclusive.

7) B `alfa = 25; bravo = -75; charlie = -64; -64 % 25 = -14`

★ANSWER KEY – CONFIDENTIAL★

8) D The `switch` matches on the `case` where `test = 4`, prints `"four"`, increments `test` to 7, prints `"eight"`, then exits the `switch-case` statement before printing the final value of `test` (7).

9) A Prints a "#" when `pound = 4`, 16, and 256. Exits the loop when `pound = 65536`.

10) E
```
b = 107 + 105 + 99 + 107 + 101 + 108 = 627
c = 107 + 109 = 216
(b + c) / 2 = (627 + 216) / 2 = 843 / 2 = 421 (integer division)
```

11) B Value of `cat`, `dog`, and output at the point of the `print()` invocation in each iteration of the loop:
```
    dog:    1 11  9  1  6
    cat:   11  9  1  6  3
 Output: -10  2  8 -5  3
```

12) B
```
sum = 15 + 30 + 60 + 120 + 240 + 480 = 945
```

13) E
```
= ((28 ^ (77 >> 3)) | (15 & 27))
= ((28 ^ 9) | (15 & 27))
= ((28 ^ 9) | 11)
= (21 | 11)
= 31
```

14) B O(1) to add new customers to the tail of the queue. O(1) to remove customers from the head of the queue.

15) C
```
longs = []
longs = [0]              (when i = 0, set bytes[0/3] = 0*2)
longs = [0, 4]
longs = [4, 4, 8]        (when i = 2, set bytes[2/3] = 2*2)
longs = [4, 4, 8, 12]
longs = [4, 8, 8, 12, 16]        (when i = 4, set bytes[4/3] = 4*2)
longs = [4, 8, 8, 12, 16, 20]
longs = [4, 8, 12, 12, 16, 20, 24]        (when i = 6, set bytes[6/3] = 6*2)
longs = [4, 8, 12, 12, 16, 20, 24, 28]
longs = [4, 8, 16, 12, 16, 20, 24, 28, 32]        (when i = 8, set bytes[8/3] =8*2)
longs = [4, 8, 16, 12, 16, 20, 24, 28, 32, 36]
```

16) B The second 1 in the string, `011001011`, breaks the pattern specified by the regular expression, `0*1(0+1+)*`.

17) D `pop()` causes an item to be removed from the stack, but `peek()` does not remove the item from the stack.

18) C Recursively produces a post-order traversal of the tree whose in-order traversal is the parameter `String s`.

19) A Recursively finds the index of parameter, `b`, in array, `a`, using binary search.

20) C Binary search yields $O(\log_2 N)$ performance in the average and worst cases.

21) D Finds the index of `'e'` in `data`. Note that `data` is not sorted, so the method actually happens upon the 2nd occurrence of `'e'` in the array.

22) B The `help()` method returns a sub-array containing the elements of array `a` from index positions `x` through `y - 1`, inclusive.

23) C $36_{10} = 2a_{13}$

24) C `slices = ["ru", "umpers"]`. The `".*"` in the regular expression is greedy and matches on all characters between the first and last `'b'` in the string (`"bber baby buggy b"`).

25) D `"2 fish"` < `"Blue fish"` < `"Red fish"` < `"one fish"` when compared lexicographically (case-sensitive).

26) C Results in an `ArithmeticException` (`"/ by zero"`) when attempting to divide `a[7] = 5` by `a[8] = 0`. The exception is caught by the first `catch()` clause (`code = 1`) and the `finally` clause is always executed (`code = 13`). Note that `ArithmeticException extends RuntimeException extends Exception`.

27) E No exceptions are thrown and the `finally` clause is always executed (`code = 3`).

28) B Outer loop iterates `one` through values of 4, 6, and 8. Inner loop iterates `ten` through values of 2, 2 through 3, and 2 through 4, respectively for each pass through the outer loop. Note that the output concatenates the values of `ten` and `one`.

★ANSWER KEY – CONFIDENTIAL★

29) D `X = (P * R) + ( P' * Q * R' )`. When the values of `P` and `R` are different from each other, the output is `0`.

Note that the correct answer choice only addresses cases in which `P != R` and makes no statement about the output if `P == R` (i.e., the output could be either `0` or `1` in that case).

30) B `((2 ^ 3) | (6 – 4)) = ((2 XOR 3) OR 2) = (1 XOR 2) = 3`

31) C `wages = 50.0 + 25.0 = 75.0`. The `paycheck()` and `comment()` methods defined in the `Worker` class bind with the `bonus` field declared in the `Worker` class (`1.25`) and never bind with either the `bonus` field declared in the `Slacker` subclass (`0.75`) or the overridden `rate` field declared in the `Employee` interface (`1.00`). But the `toString()` method in the `Slacker` subclass binds with and references the `bonus` field declared in the `Slacker` subclass (`0.75`).

32) A `wages = 50.0 + 25.0 = 75.0`

33) E `Employee` is an `interface`. It cannot be directly instantiated with `new Employee()`.

34) E

Keys	Values
Dopey	Doc
Grumpy	~~Bashful~~ Dopey
Happy	Doc
Sleepy	Sneezy

35) A `grid = [[1, 2, 3, 4], [2, 6, 7], [3, 7], [4]]`

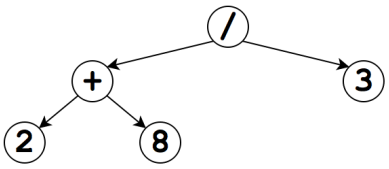
36) C

P	Q	R	X = !(R && P) && (!P && Q)	Y = (!P && Q) && (!Q || R)
0	0	0	**0**	**0**
0	0	1	**0**	**0**
0	1	0	**1**	**0**
0	1	1	**1**	**1**
1	0	0	**0**	**0**
1	0	1	**0**	**0**
1	1	0	**0**	**0**
1	1	1	**0**	**0**

37) A Post-order: `OIMENADSRLP`. In-Order: `OIAMNEPDLSR`. Level-by-level: `PALINDROMES`

38) B 111_{10} = 01101111_2; -100_{10} = 10010001_2; 1's complement of -100_{10} = 10010000_2

39)



Postfix (reverse Polish) notation: `28+3/`
Prefix (Polish) notaton: `/+283`
Infix notation: `(2+8)/3`

40) Any answer that equivalently expresses "(X Logical-AND Y) Logical-OR Z" is acceptable (use of parentheses is optional):

`XY + Z`	`YX + Z`	`Z + XY`	`Z + YX`
`(X * Y) + Z`	`(Y * X) + Z`	`Z + (X * Y)`	`Z + (Y * X)`
`(X && Y) || Z`	`(Y && X) || Z`	`Z || (X && Y)`	`Z || (Y && X)`
`(X and Y) or Z`	`(Y and X) or Z`	`Z or (X and Y)`	`Z or (Y and X)`

Conference _____ Code Number _____

UIL COMPUTER SCIENCE WRITTEN TEST

Questions (+6 points for each correct answer, -2 points for each incorrect answer)

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

FOR ADMINISTRATIVE USE ONLY

# Right:	×	6 pts	=	
# Wrong:	×	-2 pts	=	
# Skipped:	×	0 pts	=	0

	Score	Initials
Judge #1:		
Judge #2:		
Judge #3:		

Computer Science Competition
District 2 2016
Programming Problem Set

I. General Notes

1. Do the problems in any order you like. They do not have to be done in order from 1 to 12.

2. All problems have a value of 60 points.

3. There is no extraneous input. All input is exactly as specified in the problem. Unless specified by the problem, integer inputs will not have leading zeros. Unless otherwise specified, your program should read to the end of file.

4. Your program should not print extraneous output. Follow the form exactly as given in the problem.

5. A penalty of 5 points will be assessed each time that an incorrect solution is submitted. This penalty will only be assessed if a solution is ultimately judged as correct.

II. Names of Problems

Number	Name
Problem 1	Aiguo
Problem 2	Ba Tu
Problem 3	Chin
Problem 4	Daisuke
Problem 5	Ekaterina
Problem 6	Felipe
Problem 7	Gary
Problem 8	Heather
Problem 9	Isabel
Problem 10	Jakub
Problem 11	Kartik
Problem 12	Lauren

1. Aiguo

Program Name: Aiguo.java Input File: aiguo.dat

Aiguo is an avid reader and really loves grammar. He enjoys reading proofs of works, and gets to make changes to these works before they get published! However, he has noticed that it is incredibly time consuming to really highlight the changes he makes in these electronic versions. What Aiguo really wants is the ability to just make the changes he sees fit with his expertise and have a program analyze the changes he made to report to the publisher.

The changes that Aiguo makes can be represented by deletions or additions to the original text. He gets an initial work, then will either add or remove words and characters to improve and revise the works. He would like you to write a program that can identify which characters he removed and added by looking at the initial and final version of his work.

Input: The first integer will represent the number of data sets to follow. The first line of each data set will be the initial work, before Aiguo's changes. The second line of each data set will be the final work, after Aiguo's changes.

Output: Each data set should have 3 lines of output. The first, labeled "added: " will be the characters that Aiguo added to the initial text. The second, labeled "removed: " will be the characters that Aiguo removed from the initial text. The last, labeled "unchanged: " will be the characters that Aiguo did not change. Each of these should be listed in the order that the changes were made. If there exists an exchange where it was ambiguous if a correction was an additional of one word or the removal of another, choose to state the correction that comes first in the text.

Assumptions: Each work will be on one line.

Sample Input:

```
3
the brown fox curiously jumped over the sad dog
the quick brown fox jumped over the lazy dog
Ceci n'est pas ma femme.
Ceci est ma femme.
I love chocolate!
He loves chocolates!
```

Sample Output:

```
removed: curiously sd
added: quick lzy
unchanged: the brown fox jumped over the a dog
removed: n'pas
added:
unchanged: Ceci est ma femme.
removed: I
added: Hess
unchanged:  love chocolate!
```

2. Ba Tu

Program Name: BaTu.java Input File: batu.dat

Ba Tu has just learned to play Sudoku, but still needs help figuring out what values any particular square could be. The rules are that every row, column, and every 3X3 sub-box within the large grid have unique values from 1 to 9.

Your job is to help Ba Tu get started with any particular empty box, showing him what digit, or digits, could possibly work for that empty box.

In the puzzle shown here, the empty blank at position 1,1 at the top left corner of the puzzle cannot be a 5, 1, or 4 since that row already contains those numbers, and it cannot be 2, 3, or 9 since that column contains those, and it cannot be a 7 since there is a 7 in the sub-box for that blank. That means that 6 or 8 would work in that blank.

_ 5 _ 1 _ 7 _ _ _	_ 1 _ _ _ _ 9 _ 5	_ 4 _ 6 _ 2 _ _ _
2 _ 8 _ 4 _ 9 _ 1	_ 3 _ _ 7 _ _ 8 _	5 _ 1 _ 2 _ 4 _ 6
_ _ _ 3 _ 4 _ 2 _	4 _ 1 _ _ _ _ 6 _	_ _ _ 7 _ 9 _ 1 _

For the blank at row 4, column 2, between the 2 and 8, the only values that could work are 6 and 7, since that row already has the digits 1, 2, 3, 5 and 8, the column also has 4, and that sub-box also contains the 9.

Input: Nine strings of nine digits, each on a separate line, representing the beginning values of a 9X9 Sudoku puzzle. A zero digit represents an empty blank. After the initial nine strings will be several pairs of integers representing the column and row of an empty blank, each pair on a separate line.

Output: All the possible digits that could be placed in the empty blank indicated by each pair of digits. The digits are to be listed in ascending order with no spaces between.

Sample input:

```
050010040
107000602
000905000
208030501
040070020
901080406
000401000
304000709
020060010
1 1
4 2
2 5
3 5
```

Sample output:

```
68
67
4
24
```

3. Chin

Program Name: Chin.java Input File: chin.dat

In calculus, there exists an operation called a derivative. This operation will tell you the slope of the curve at a given point. The rules for finding the derivative for a polynomial are very well defined, and rather formulaic. Chin is getting behind on his calculus homework, and he would like you to write an evaluator, that given an equation and the point at which to evaluate it, determines the slope at that point. This means, you will have to take the derivative of the polynomial, as defined by the rules below, and evaluate the new equation produced by taking the derivative, at the point given.

The rules for determining the derivative are as follows:

1. if there is a term without an x in it, remove it
2. if there is a multiplication, e.g., ***a * x***, keep only the coefficient, e.g., ***a***. (Note, x / 7 is equivalent to (1 / 7.0) * x)
3. if there is an exponent, e.g., ***a * x^b***, multiple the x by the exponent, and reduce the value of the exponent by 1, e.g., ***a * b * x ^ (b-1)***.

Following these rules in the equation: **5 * x^4 + 3 * x + 2** gives us:

By rule 1, the "**+ 2**" at the end is removed. By rule 2, "**3 * x**" becomes "**3**". By rule 3, the "**5 * x^4**" becomes "**5 * 4 * x^3**".

All of Chin's equations will be exponents (^), multiplication (*), divisions (/), subtractions (-), and additions (+). All coefficients and exponents will be integers. There will be no parenthesis or brackets and typical order of operations do apply.

Input: The first integer will represent the number of data sets to follow. The first line of the dataset will be the polynomial Chin wants you to take the derivative of. The second line of each data set will be one floating point number that will be the point (value of x) at which to evaluate the expression derived.

Output: For each output, display the value of the derivative of the given polynomial evaluated at the given point, rounded to two decimal places.

Assumptions: All multiplications, additions, subtractions, and divisions will be separated by a space. All values will be positive.

Sample Input:

```
3
5 * x^2 + x / 7 + 9
1372.3
7 * x^3 + 7 * x + 15
100.1
1 / 7 + 17 + 9 * x + 15 * x^3
0.0
```

Sample Output:

```
13723.14
210427.21
9.00
```

4. Daisuke

Program Name: Daisuke.java **Input File: daisuke.dat**

Daisuke loves to create symmetrical character patterns, like the one shown below. Your job is to write a program that outputs this exact example of his work.

Input: NONE

Expected output:

```
\"""/
%\''/%
%.\/.%
%./\.%
%/''\%
/"""\
```

5. Ekaterina

Program Name: ekaterina.java **Input File: ekaterina.dat**

Ekaterina has just read the classic Jules Verne story **Around The World In Eighty Days**, where the main character Phileas Fogg claimed he could do the title task, making a huge bet with some of his friends, and then proceeding to head east, on his journey around the world. Phileas was a fanatic about time, and was always checking the time and measuring how far he had come, and what time it was in London, his demarcation point, and ultimate destination.

He would also try and predict what time it was in future destinations. For example, if he was in London, he would be in the zero time zone, otherwise known as Greenwich Mean Time, based on the location of the Royal Observatory in Greenwich, a suburb of London. If he wanted to know the current time in France, all he would do is add an hour, since France was in the next time zone to the east.

Currently all of the UK, Portugal, and parts of West Africa, are in the GMT zone. As you head East into continental Europe and beyond, you pass into other time zones, each one roughly 15 degrees to the East, with 360 degrees representing a full circumnavigation of the globe.

Each new time zone eastward has a time difference of one hour more than the previous one. For example, France, Germany and Italy all are at 1 PM if Big Ben in London strikes noon. Beyond that into Eastern European countries, the next zone shows 2 PM, and so on, around the world.

Heading west, the same thing happens, except in reverse. Each time zone 15 degrees further west is another hour earlier. For example, New York City is 5 time zones to the west, 5 hours earlier, so when it is 7AM in New York, it is already noon in London.

A number line to show this would have zero in the middle, indicating Greenwich time, 15E starting the time zone containing Western Europe, 30E starting the Eastern European zone, and so on. Halfway around, at 180 degrees, is the International Date Line.

```
                                NY              GMT
IDL 165W 150W 135W....90W 75W 60W 45W 30W 15W 0 15E 30E 45E...135E 150E 165E IDL.
```

Phileas would have appreciated a computer program to help him keep track of all of this, like the one Ekaterina wants to write to solve this problem, but needs your help.

Input: Several data sets, each on one line, consisting of four items. The first indicates the starting location, with a degree value between 0 and 180, inclusive, followed by the letter 'W' or 'E', the second a three letter abbreviation for the current day of the week, the third the two digit time of the current location, followed by "AM" of "PM", and the fourth the remote time zone, using the same format as the first data item.

Output: The day and time at the remote location, in the format HHXM, with HH meaning the two-digit time and the X either the letter 'P' or 'A'.

Sample input:

```
20E WED 01PM 0E
65W FRI 07AM 5E
4E SAT 06AM 23E
157W MON 07AM 110W
170E SAT 06PM 140W
```

Sample output:

```
WED 12PM
FRI 12PM
SAT 07AM
MON 10AM
FRI 09PM
```

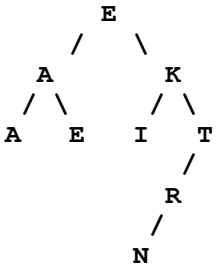
6. Felipe

Program Name: Felipe.java Input File: felipe.dat

Felipe is fascinated with binary search trees, especially with the four different traversals he just learned in class: inorder, preorder, postorder, and reverse order.

According to the rules he learned in class about building a character tree using a word, he practices doing this with his friend's names, and especially enjoys using his beautiful girlfriend's equally beautiful name, EKATERINA, with whom he is head-over-heels in love!

He starts with the first letter, E, as the root, and then sees that the next letter, K, is alphabetically after E, so he inserts it as a right node to the E, and then the A as a left node to the E. When he gets to the T, he goes right at the E, and then right again at the K, and proceeds on through the rest of the letters of her name, resulting in the binary search tree shown below. He decides to allow duplicate letters and slides those to the left as they reach a matching node.



He then proceeds to traverse in order, which means start at the top of the tree, going as deep to the left along each branch, and only outputting a node when reaching the end of a branch (a leaf node) or outputting a node after its left branch has been completely processed.

The inorder traversal (alpha order) of this tree would be: **AAEEIKNRT**
The preorder traversal (output a node BEFORE traversing either the left or right subtrees) would be: **EAAEKITRN**
Postorder traversal (output after both branches are completely traversed): **AEAINRTKE**
The reverse order traversal is just the inorder traversal backwards, which Felipe thinks about for a bit, and then realizes what to do, resulting in: **TRNKIEEAA**

Input: Several uppercased names containing no other symbols , each on one line, and each followed by a single character indicating the traversal to be performed: **E**(preorder), **O** (postorder), **I** (inorder), **R** (reverse order).

Output: The indicated traversal for each name, as shown in the examples.

Sample Input:
```
EKATERINA I
EKATERINA E
EKATERINA O
EKATERINA R
```

Sample Output:
```
AAEEIKNRT
EAAEKITRN
AEAINRTKE
TRNKIEEAA
```

7. Gary

Program Name: Gary.java Input File: gary.dat

Gary has just encountered boolean postfix expressions in class, and is a bit confused. He understands the easy ones, like, "true and false" becomes "true false and", and knows how to evaluate them according the following rules of thumb:

- "and": evaluates to "false" only when both operands are "true"
- "or": evaluates "false" only when both operands are "false"
- "xor": evaluates to "true" only when the operand values are opposite
- "not": reverses the value from "true" to "false", or vice versa

It's evaluating the more complex ones in postfix form that give him a bit of trouble.

For example, "true or false and true" is confusing to him. "*What is the postfix version?*", he wonders, "*and then what is the answer?*" Well, his teacher helps him out and just gives him the postfix version, for now, which is "true false true and or", and then shows him how to process it.

His teacher says, "*Take the first two boolean values and see if there is an operator immediately after them. If there is, do that operation, and replace the result in place of that expression. If not, ignore the first operand for now, and look at the next pair to see if there is an operator following, and evaluate it and replace it with an answer.*"

"*In the example, "true false true and or", there is no operator for the first two operands, so you ignore the first operand and look at the next two, which have an "and" after them. Well, "false and true" evaluates to "false", so you replace the "false true and" with just "false", which now gives you the expression, "true false or", which is "true", and is the final value of the expression.*

"*If you have a "not" in the expression, like this one, "true false or not", the result here would first be, "true not", since "true false or" becomes "true", and then "false", since the "not" operator only needs one operand, and "not true" means "false".*"

Input: Several lines of data, each line with a postfix boolean expression made up of the words, "true", "false", "not", "and", "or", and "xor", with no parentheses, all separated by single spaces.

Output: The final **true** or **false** value of the postfix boolean expression.

Sample input:

```
true true or
true false and
true false true and or
true false or not
```

Sample output:

```
true
false
true
false
```

8. Heather

Program Name: Heather.java **Input File: heather.dat**

Heather has just learned that it is possible to have any base you want, other than base ten, two, eight, and sixteen. She figures since there are 10 counting digits and 26 letters of the alphabet, you could easily represent a value up to base 36 if you wanted to.

She decided to make up an exercise where two values were represented, each value in a different base, anywhere from base 2 to base 36, where three of the four value/base items were known, and the fourth unknown. She then proceeded to try and find the missing item.

For example, if she knew that 25 base 10 was equal to *"something"* in base 8, she would do some math and determine that 31 was the missing value, and she would write the original four items in the order 25 10 X 8, with X indicating the missing value.

On the other hand, she could write the expression 25 X 31 8, and try to find the correct base that has 25 as the value, and is equal to 31 base 8. The answer, of course, is 10.

The expression 25 10 31 X, would result in 8 as the missing value.

She then tried something really strange, using X 23 FACE 16, wondering if she could figure this one, which she did by converting FACE, base 16 to base 10, and then to base 23, with a final answer of 568D.

Given four items N1 B1 N2 B2 in an expression, with any of the four items replaces with an X, write a program to find the missing value of X in each expression.

Input: Several sets of data, each consisting of four items representing two pairs of value/base pairs, with one of the four items indicated by the letter X, representing a missing value for one of the values, or one of the bases.

Output: The missing value for each data set, with any alphabetic characters shown in uppercase.

Sample input:

```
25 10 X 8
25 X 31 8
25 10 31 X
X 23 FACE 16
A5 X BC 13
```

Sample output:

```
31
10
8
568D
15
```

9. Isabel

Program Name: Isabel.java Input File: isabel.dat

In computer science class, Isabel has been studying binary numbers, and has made up a game to practice her binary skills. She understands the binary place value system, where the first (rightmost) column is worth 1, the second worth 2, and then 4, 8, and so on. She has decided to label these columns A, B, C, etc., with A being the label for the rightmost ones place, B the 2s place, C the 4s place, etc. It helps her to see the whole picture when she arranges the numbers in column format, like this:

```
Base Two    Base ten
C B A
0 0 0 =     0
0 0 1 =     1
0 1 0 =     2
0 1 1 =     3
1 0 0 =     4
1 0 1 =     5
1 1 0 =     6
1 1 1 =     7
```

She sees that for this base ten range of values from 1 to 7, the number 6 has 1s in columns B and C. The numbers that all have 1s in the A column are 1, 3, 5, and 7. The numbers that have 1s in the B column are 2, 3, 6, and 7. In the C column, 1s correspond to the values 4, 5, 6 and 7.

Her game goes like this. She picks a random number between 1 and 500 representing the range of values within which her mystery number will be. She then picks a random letter that would represent one of the columns in the binary form of the maximum number in her range, and then one or more other letters, also column labels within the range of the maximum number's binary form. She then tries to figure out the resulting number, where there would be a 1 in each of the binary columns represented by the letters she picked. After she gets the number, she then counts how many other values have a 1 in the column of the first letter she picked.

For example, if she picked 7 as the maximum value in her mystery number range, and then picked first the letter B, as well as another letter C, the binary form of the mystery number would be 110, since there is a 1 in both the C and B columns but not the A column. This number converted to base ten is the value 6, her mystery number. The other numbers between 1 and 7 that also have a 1 in the B column (the first letter she picked) of their binary forms are 2, 3, and 7, making a total of 4 values that have 1s in the B column within that range.

In another example, with 15 as the maximum range value, and only the letter A picked, the resulting binary form of the mystery number only has a 1 in the A column, making 0001, which is equal to the decimal value 1. Other numbers from 1 to 15 that also have a 1 in the A column are the values 3, 5, 7, 9, 11, 13, and 15, for a total of 8 values.

Input: Several data sets, each on one line with single space separation, consisting of an integer M for the maximum range value ($1<M<=500$), an integer N for the number of letters to follow, and then N capital letters. The letters given will correspond to the binary place values, with A always indicating the ones place, B the 2s place, C the 4s place, D, E, F, and so on.

Output: Two integer values, the first of which is the base ten mystery number represented by 1s in the input letter column labels, as described above, and then a value representing the number of integers within the given range, whose binary form contains a 1 in the column matching the first letter listed in the data set.

Assumptions: All letters in each data set will be unique, and the number and nature of the letters given will not exceed the number of columns required for the binary form of the maximum value indicated.

Sample input:

```
7 2 B C
15 1 A
10 2 B A
130 4 F D E G
```

Sample output:

```
6 4
1 8
3 5
120 64
```

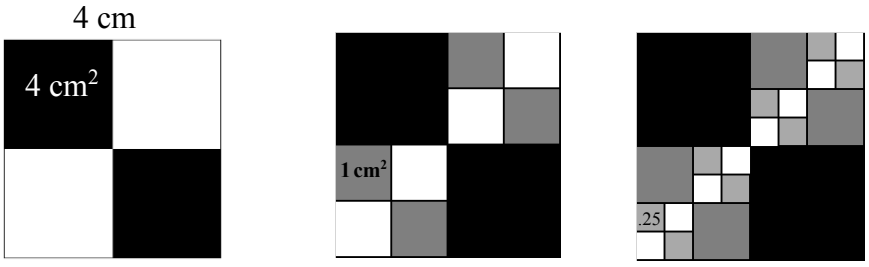
10. Jakub

Program Name: Jakub.java Input File: jakub.dat

Jakub is quite the graphic artist and has been experimenting with drawing square-based patterns, dividing a square into four parts, coloring two diagonal squares, and then dividing the other two smaller squares into four parts, coloring two, dividing the other two, and so on until the divided square reaches a size smaller than 1 centimeter, where they become too small to divide anymore.

His analytical mind wonders what the area is of the colored squares, and sets out to do the math, but has some trouble figuring this out. Please help Jakub calculate how much are of his repeatedly divided square is colored in.

For example, he might start with a 4 cm square, like the one shown below, divide it into four 2X2 squares, each with an area of 4 cm^2, two of which he colors in for a total of 8 cm^2. The two remaining squares are then divided into four 1 cm squares, adding 4 cm^2 to the colored area total, for a total of 12 cm^2, and the remaining 1 cm squares are each divided into squares measuring 1/2 by 1/2, adding a total area of 2 cm^2 more (8 times 0.25 cm^2), for a total of 14 cm^2 of colored-in area.



In another example using a 5 cm square, the total colored-in area would be 21.875 (6.25 + 6.25 + 1.5625 + ….).

Input: Several integers N (1<=N<=100), each on one line, each representing the side length of a square.

Output: The total area for each NXN square colored in using the process described above, rounded to three decimal places of precision.

Sample input:

```
4
5
24
16
```

Sample output:

```
14.000
21.875
558.000
248.000
```

11. Kartik

Program Name: Kartik.java Input File: kartik.dat

Kartik loves to mess around with his friend's names, as well as his own, and has come up with several processes to do this. For example, he may take the leftmost character of his name, delete it, and put a dash on the other end. He calls this a left shift 1, or LS-1, for short. He would notate this as: **LS-1 KARTIK**, and would write the result as:

KARTIK ==> ARTIK-

Similarly, he would do a Right Shift, followed by a number, and call it RS-N, with N being some integer value, no longer than the name he is dealing with. He would make sure all of the numbers he chooses for any of his processes don't go beyond the number of letters in the name he is messing with.

He also does a "circulate", which works like the "shift" operation, except that he inserts each removed letter on the other end instead of putting a dash. He can circulate the letters, either to the right or to the left, with an N value indicating how many times to circulate, but again no greater than the length of the name he is messing with. His notation for this is either RC-N, or LC-N. For example, if he did RC-2 AIGUO, the result would be:

AIGUO ==> UOAIG

where the rightmost two letters of the name are circulated back to the front.

Another circulate process he uses is Mid Circle, and he uses four items to control this one: S, indicating a starting position in the name, 1 being the first letter of the name, L being a number that indicates how many letters of the name to be circulated, starting from S, X indicating how many times to circle the letters, and finally the character R or L to indicate the direction of the circulate. For example, MC-243R EKATERINA would take "KATE", which is 4 letters starting at position 2, and circulate those letters 3 times to the right, which would make "ATEK", and thus the full effect would be:

EKATERINA ==> EATEKRINA

The last process he likes to use is a reverse, but not of all of the letters, but just a few somewhere in the name. For example, if he applied a REV-42 to his friend's name, DAISUKE, he would start with the fourth letter, an S, taking 2 letters from there, "SU", reversing them within the name, resulting in:

DAISUKE ==> DAIUSKE

What he really likes to do is combinations. For example, RS-2 REV-24 HEATHER, would first produce HHTAEER with the reverse, and then --HHTAE. Sometimes he goes crazy and puts several combinations (no more than 6) with a name, always doing the one closest to the name first, and then working backwards until all have been completed.

Input: Several lines of data, each on one line, each consisting of a process, or series of processes, ending with a name, all in caps. Single spaces separate each item in the data line.

Output: The original name, followed by **" ==> "**, and then the resulting name after all processes have been completed.

Sample input:

```
LS-1 KARTIK
RC-2 AIGUO
MC-243R EKATERINA
REV-42 DAISUKE
RS-2 REV-24 HEATHER
```

Sample output:

```
KARTIK ==> ARTIK-
AIGUO ==> UOAIG
EKATERINA ==> EATEKRINA
DAISUKE ==> DAIUSKE
HEATHER ==> --HHTAE
```

12. Lauren

Program Name: Lauren.java Input File: lauren.dat

Lauren loves to travel. She just got her brand new passport, and she is excited to travel the world. She is overwhelmed by the number of cities she wants to visit. To be the most efficient, she wants to be sure to visit every city on her list, but she wants to make sure to do it covering the least amount of distance as possible. Oh, and she doesn't really care about getting back home after reaching the last city…at least for a while.

Input: The first integer will represent the number of data sets to follow. The next line will have an integer representing the number of cities she wants to visit on this trip, followed by a list of cities, each on one line and will contain no spaces. The first city in this list represents the city that Lauren is currently in. The next line will be an integer representing the number of available flights. The following lines will be two cities, separated by a space, and an integer representing the number of miles that flight covers.

Output: For each data set, print the names of the cities in the order that she should visit them. She must start in the city that she is currently in. Each flight is represented by a "=>" sign between the cities.

Assumptions: There will always be a flight to the city Lauren wants to visit. If there is a flight from city A to city B there is assumed to be the same flight from city B to city A. If there is no flight listed between two cities it is assumed she cannot fly between them. There will be at least two cities to travel between. There will always exist a route between any two cities, although it may take several hops. The distance between any two adjacent cities will be greater than 1 mile. The best solution will never require Lauren to backtrack.

Sample Input:

```
3
3
SanFrancisco
NewYork
LosAngeles
3
SanFrancisco LosAngeles 383
SanFrancisco NewYork 2905
LosAngeles NewYork 2789
5
Austin
Istanbul
Calcutta
NewYork
London
10
Austin Istanbul 6500
Austin Calcutta 8773
Austin NewYork 1513
Austin London 4921
Istanbul Calcutta 3652
Istanbul NewYork 5020
Istanbul London 1556
Calcutta NewYork 7929
Calcutta London 4853
NewYork London 3465
4
Reykjavik
Aarhus
Geneva
Budapest
5
Reykjavik Aarhus 1515
Reykjavik Geneva 2379
Reykjavik Budapest 2443
Aarhus Budapest 933
Geneva Budapest 787
```

Sample Output:

```
SanFrancisco => LosAngeles => NewYork
Austin => NewYork => London => Istanbul => Calcutta
Reykjavik => Aarhus => Budapest => Geneva
```

UIL COMPUTER SCIENCE WRITTEN TEST

2016 REGION

APRIL 21-23, 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.
  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)
```

UIL COMPUTER SCIENCE WRITTEN TEST – 2016 REGION

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 $3D_{16}$ * 13_8?

A) 101001111_2 **B)** 22131_4 **C)** 1327_8 **D)** 761_{10} **E)** $29F_{16}$

Question 2.

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

A) 2 **B)** 5 **C)** 5.0 **D)** 6 **E)** 6.4

```
double m = 0.4 + 1.2 * 8;
out.println(m / 2);
```

Question 3.

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

A) (002016) **B)** (00002016) **C)** (-0002016)

D) -0002016 **E)** -00002016

```
int year = 2016;
out.printf("%(08d", -year);
```

Question 4.

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

A) .0100100 ..100100 .0100100

B) .0.00.00 .... .0.00.00

C) .0100100 1.100100 10100100

D) 0.00.00

E) .0.00.00 1.1..1.. 10100100

```
String mixed = "10100100";
out.print(mixed.replace("1", "."));
String zeros = mixed;
String ones = mixed.replace("0", ".");
out.println(" " + ones + " " + zeros);
```

Question 5.

Which of the following is equivalent to the Boolean expression on the right assuming that w, x, y, and z have been initialized with integer values?

A) w <= x || y != z

B) w >= x || y == z

C) w > x && y == z

D) w > x || y == z

E) !(w <= x) && !(y != z)

```
!(w <= x && y != z)
```

Question 6.

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

A) 0.111

B) 0.125

C) 6.000

D) 8.000

E) 9.000

```
double raw = -10.0 / 4;
double floor = Math.abs(Math.floor(raw));
double ceil = Math.abs(Math.ceil(raw));
out.printf("%.3f", Math.pow(floor, ceil));
```

Question 7.

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

A) 83 **B)** 236 **C)** 362 **D)** 623 **E)** 632

```
int hund = 236;
int ten = hund / 10;
int one = hund % 10;
out.println(ten + 10 * one);
```

Question 8.

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

```
int iffy = 12345;
```

A) A **B)** B **C)** C **D)** D **E)** E

```
if (iffy / 9 > 1000)
  if (iffy * 4 > 50000)
    out.print("A");
  else
    out.print("B");
else
  if (iffy % 3 == 0)
    if (iffy % 5 == 0)
      out.print("C");
    else
      out.print("D");
  else
    out.print("E");
```

Question 9.

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

A) 124862480 **B)** 12486248

C) 124862486 **D)** 1248624–80

E) The code segment prints an infinite string of digits.

```
byte digits = 1;
do {
  out.print(digits % 10);
  digits *= 2;
} while (digits < 128);
```

Question 10.

What is the return value of the following invocation of the `get()` method from a client class?

```
int[] q = {7, 1, 3, 4, 9, 8, 2, 5, 0, 9};
out.println( get(q, 7, 1) );
```

A) –1 **B)** 2 **C)** 3 **D)** 5 **E)** 8

```
public int get(int[] x, int y, int z) {
  int w = -1;
  for (int i = y - 1; i >= z; i--) {
    if (x[i - 1] > x[i + 1]) w = x[i + 1];
  }
  return w;
}
```

Question 11.

Assuming that the text file, `seuss.txt`, contains the values shown to the right, what is the output of this code segment?

A) 1

B) 2

C) 3

D) 4

E) No output due to an error.

seuss.txt

```
One fish
Two fish
Red fish
Blue fish
```

```
List<String> fish = new ArrayList<>();
Scanner fin = new Scanner("seuss.txt");
while (fin.hasNextLine()) {
  fish.add(fin.nextLine());
}
out.println(fish.size());
```

Question 12.

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

A) 1.0 **B)** 3.5 **C)** 4.0

D) 7.875 **E)** 8.0

```
double[] eight = new double[8];
double octo = 0;
for (int i = 0; i < eight.length; i++)
  eight[i] = i / 8.0;
for (double ocho : eight)
  octo += ocho;
out.println(octo);
```

Question 13.

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

A) –90 **B)** –30 **C)** 80 **D)** 85 **E)** 124

```
int me = 5;
int you = 24;
int us = 3;
out.print(me - you / us * me + you * me);
```

Question 14.

Which of the following Java classes does NOT implement the `Comparable` interface?

A) Random **B)** String **C)** Boolean **D)** File **E)** Integer

Question 15.

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

A) 1:1 **B)** 0:3 **C)** 1:3 **D)** 2:3

E) No output due to an error.

```
List<List<Object>> all = new ArrayList<>();
List<Object> some = new ArrayList<>();
some.add(all);
some.add(all.size());
all.add(some);
some.add(some.size());
out.println(all.size() + ":" + some.size());
```

Question 16.

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

A) `false null` **B)** `null null`

C) `false false` **D)** `false true`

E) `true null`

```
boolean[] bool = new boolean[10];
Boolean[] Bool = new Boolean[10];
out.println(bool[2] + " " + Bool[2]);
```

Question 17.

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

A) `-2147483648` **B)** `-1`

C) `0` **D)** `1`

E) `2147483647`

```
int max = Integer.MAX_VALUE;
int min = Integer.MIN_VALUE;
int sum = (-max) + (-min);
out.println(sum);
```

Question 18.

Which of the following could replace **<#1>** in the code segment to the right to initialize `posneg` to a value of either `-1` or `1`?

A) `(int)(Math.random() * 4 - 1)`

B) `(int)(Math.random() * 2) - 1`

C) `(int)(Math.random() * 2) * 2 - 1`

D) `(int)(Math.random() * 2) * -1`

E) More than one of these.

```
int posneg = <#1>;
```

Question 19.

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

A) `[]`

B) `[, -, , --, , , ----]`

C) `[.-., --.....----.]`

D) `[.-., ., --., .., ---, -.]`

E) `[, -, --, ----]`

```
String R = ".-.";
String E = ".";
String G = "--.";
String I = "..";
String O = "---";
String N = "-.";
String morse = R + E + G + I + O + N;
String[] dashes = morse.split(E);
out.println(Arrays.toString(dashes));
```

Question 20.

What return value is printed after the following invocation of the `find()` method from a client class?

```
int[] bits = {1, 0, 1, 0, 1, 0, 0, 1};
out.println( find(bits, 0) );
```

A) `-1` **B)** `1` **C)** `3` **D)** `5` **E)** `6`

```
public int find(int[] data, int item) {
  int i = -1;
  for (int j = 0; j < data.length; j++) {
    if (data[j] == item)
      i = j;
  }
  return i;
}
```

Question 21.

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

A) `2nd first FOURTH Third`

B) `2nd FOURTH Third first`

C) `FOURTH Third first 2nd`

D) `first 2nd Third FOURTH`

E) `first FOURTH Third 2nd`

```
Queue<String> queue = new PriorityQueue<>();
queue.add("first");
queue.add("2nd");
queue.add("Third");
queue.add("FOURTH");
while (!queue.isEmpty()) {
  out.print(queue.remove() + " ");
}
```

Question 22.

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

A) `2` **B)** `15` **C)** `33` **D)** `62` **E)** `124`

```
byte scan = 31;
scan <<= scan / 15;
out.println(scan);
```

Question 23.

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

A) `15` **B)** `37` **C)** `52` **D)** `77` **E)** `112`

```
out.println(Integer.parseInt("52", 15));
```

Question 24.

What is the output of line **<#1>** in the **Client Code** to the right?

A) 9 **B)** 14 **C)** 15 **D)** 21 **E)** 36

Question 25.

What is the output of line **<#2>** in the **Client Code** to the right?

A) [E, G, I, O, N, R]

B) [N, O, I, G, E, R]

C) [E, G, I, N, O, R]

D) [R, E, G, I, O, N]

E) [R, O, N, I, G, E]

Question 26.

Which of the following algorithms is implemented by the `process()` method to the right?

A) Sequential Search **B)** Merge Sort

C) Insertion Sort **D)** Selection Sort

E) Quicksort

Question 27.

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

A) $O(\log_2 N)$ **B)** $O(N)$ **C)** $O(N * \log_2 N)$

D) $O(N^2)$ **E)** Indeterminate

```
static int process(List<String> a) {
  int n = 0;
  for (int i = 0; i < a.size(); i++) {
    n += help(a, i);
  }
  return n;
}

static int help(List<String> a, int i) {
  String c = a.get(i);
  int n = i - 1;
  while (n >= 0) {
    if (a.get(n).compareTo(c) > 0) break;
    n--;
  }
  a.add(n + 1, a.remove(i));
  return i - n - 1;
}
```

Client Code

```
String str = "REGION";
List<String> c = new ArrayList<>();
for (int i=0; i<str.length(); i++)
  c.add(str.substring(i, i+1));
out.println(process(c));       //<#1>
out.println(c);                //<#2>
```

Question 28.

Which of the following strings does NOT match the regular expression to the right?

A) UIL **B)** uil2016

C) 2016 Regional **D)** (2016)

E) uil2016regional

```
([a-z]*[0-9]+)*.+[^0-9]
```

Question 29.

What return value is printed after the following invocation of the `hash()` method from a client class?

```
out.println(hash("abcdefghij", 3));
```

A) fgc **B)** ghd

C) hid **D)** ije

E) No output due to an error.

Question 30.

What return value is printed after the following invocation of the `hash()` method from a client class?

```
out.println(hash("1234567890", 2));
```

A) 673 **B)** 784

C) 894052 **D)** 90563

E) No output due to an error.

```
public String hash(String src, int n) {
  if (n > src.length()) return "";
  String a = hash(src, n * 2);
  String b = hash(src, n * 2 + 1);
  return a + b + src.substring(n, n + 1);
}
```

Question 31.

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

A) −8531 **B)** −1 **C)** 0 **D)** 8531 **E)** 57005

```
int dead = 0xdead;
int alive = ~dead;
int wanted = dead ^ alive;
out.println(wanted);
```

Question 32.

What is the output of line **<#1>** in the **Client Code** to the right?

A) 2T 3T 4T 5T 6T 7T 1T

B) 1T 2T 3T 4T 5T 6T 7T

C) 1T

D) 7T 6T 5T 4T 3T 2T 1T

E) 1T 7T 6T 5T 4T 3T 2T

Question 33.

What is the output of line **<#2>** in the **Client Code** to the right?

A) 1T

B) 2T 3T 4T 5T 6T 7T 1T

C) 1T 7T 6T 3T 2T

D) 2T 3T 6T 7T 1T

E) 1T 2T 3T 4T 5T 6T 7T

Question 34.

What is the output of line **<#3>** in the **Client Code** to the right?

A) 2H 3H 6T 7T 1T

B) 1T 7T 6H 3H 2T

C) 2T 3H 6H 7T 1T

D) 1T 2T 3H 4H 5T 6T 7T

E) 1T 7T 6H 5T 4T 3H 2T

Question 35.

What is the output of line **<#4>** in the **Client Code** to the right?

A) 4T 5T

B) 4H 5T

C) 2T 3H 4T 5T 6H 7T 1T

D) 1T 2T 3H 4H 5T 6T 7T

E) 2T 3H 6H 7T 1T

Question 36.

What type of data structure does the `Disc` class to the right model?

A) Stack

B) Linked List

C) Hash Table

D) Queue

E) Priority Queue

```
public class Disc {
  boolean state;
  private Disc east, west;
  private int i;
  static int n;

  public Disc() { i = ++n; }

  public Disc(Disc w, Disc e) {
    this();
    bind(w, this);
    bind(this, e);
  }

  public static void bind(Disc w, Disc e) {
    w.east = e;
    e.west = w;
  }

  public boolean flip() {
    if (state && east != null)
      east.state = !east.state;
    if (!state && west != null)
      west.state = !west.state;
    return state = !state;
  }

  public Disc get(int x) {
    Disc d = this;
    while (d.east != null) d = d.east;
    do { d = d.west; }
    while (d != null && d.i != x);
    return d;
  }

  public String toString() {
    String s = "";
    Disc d = this;
    while (d.east != null) d = d.east;
    do {
      s += d.i + (d.state?"H ":"T ");
      d = d.west;
    } while (d != null);
    return s;
  }
}
```

Client Code

```
Disc base = new Disc();
Disc disc = new Disc();
Disc.bind(base, disc);
for (int i = 0; i < 5; i++)
  disc = new Disc(base, disc);
out.println(base);              //<#1>
Disc dFour = base.get(4);
Disc.bind(base.get(6), disc.get(3));
out.println(base);              //<#2>
base.get(3).flip();
out.println(disc);              //<#3>
out.println(dFour);             //<#4>
```

Question 37.

Consider the adjacency matrix to the right that describes a connected graph of 7 nodes. A "0" in any cell indicates that there is no direct connection between two nodes. A "1" indicates that there is a path from the corresponding node for the row to the corresponding node for the column. How many unique paths are there from node A to node D that visit each node at most once per path?

A) 1

B) 2

C) 3

D) 4

E) 5

	A	B	C	D	E	F	G
A	0	1	0	0	0	1	1
B	1	0	1	0	0	0	0
C	0	0	0	0	0	0	1
D	1	1	0	1	0	0	0
E	0	0	1	0	1	0	0
F	0	0	0	0	1	0	0
G	0	1	0	1	1	1	0

Question 38.

Which of the following is the equivalent Reverse Polish Notation (RPN) of the arithmetic expression to the right?

A) `3 7 2 / + 4 5 * –`

B) `5 * 4 – 2 / 7 + 3`

C) `– + * 3 / 4 5 7 2`

D) `– + 3 / 7 2 * 4 5`

E) `3 7 2 / 4 5 * + –`

```
3 + 7 / 2 – 4 * 5
```

Question 39.

Write a simplified, Boolean expression to describe output X, given inputs A, B, C, and D, 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	D	**X**
0	0	0	0	**0**
0	0	0	1	**0**
0	0	1	0	**0**
0	0	1	1	**0**
0	1	0	0	**0**
0	1	0	1	**1**
0	1	1	0	**0**
0	1	1	1	**1**
1	0	0	0	**0**
1	0	0	1	**0**
1	0	1	0	**1**
1	0	1	1	**1**
1	1	0	0	**0**
1	1	0	1	**1**
1	1	1	0	**1**
1	1	1	1	**1**

Question 40.

Given the two 8-bit, signed, 2's complement binary representations in the expression to the right, what is the decimal value of the 8-bit, signed, 2's complement binary representation that results from evaluating the expression?

Write your answer on the answer sheet.

$$11011101_2 + 01011101_2$$

★ DOUBLE-CHECK YOUR ANSWERS ★

★ANSWER KEY – CONFIDENTIAL★

UIL COMPUTER SCIENCE WRITTEN TEST – 2016 REGION

Questions (+6 points for each correct answer, -2 points for each incorrect answer)

1) E	11) A	21) B	31) B
2) C	12) B	22) E	32) A
3) A	13) D	23) D	33) D
4) E	14) A	24) A	34) C
5) D	15) C	25) E	35) E
6) E	16) A	26) C	36) B
7) A	17) D	27) D	37) C
8) B	18) C	28) B	38) A
9) E	19) A	29) B	*39) (A*C) + (B*D)
10) C	20) E	30) E	40) 58

** See "Explanation" section below for alternate, acceptable answers.*

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.

Explanation

1) E $3D_{16}$ * 13_8 = 1010011111_2 = 22133_4 = 1237_8 = 671_{10} = $29F_{16}$

2) C `(0.4 + 1.2 * 8) / 2 = (0.4 + 9.6) / 2 = 10.0 / 2 = 5.0`

3) A `"%(08d"`: Negative values are to be formatted with parentheses instead of a negative sign and the parentheses count toward the 8-character field width. Number should be zero-padded to fill out the remaining field width.

4) E The `replace()` method creates a new `String` in which each substring that matches the literal target sequence (1st parameter) is replaced with the specified literal replacement sequence (2nd parameter). The original `String` is not modified. Hence, `mixed` is never actually modified in the code segment, `zero` is a reference to the unmodified `mixed`, and `ones` references a modification of `mixed`.

5) D `!(w <= x && y != z) = !(w <= x) || !(y != z)`, by DeMorgan's Law.

6) E `raw = -2.5; floor = 3.0; ceil = 2.0; Math.pow(3.0, 2.0) = 9.0`

7) A `hund = 236; ten = 23; one = 6; 23 + (10 * 6) = 83`

8) B `(12345 / 9) = 1371 > 1000; (12345 * 4) = 49380 < 50000`

9) E Variable `digits` overflows when its value exceeds `127` (i.e., `Byte.MAX_VALUE`). `128` overflows to become `-128`. `-256` underflows to become `0`. The resulting values for the variable `digits` at the start of each iteration are `1, 2, 4, 8, 16, 32, 64, -128, 0, 0, 0, 0, ...` causing the output to be `"1248624-80000..."` (Infinitely repeating zeroes).

★ANSWER KEY – CONFIDENTIAL★

10) C For each index position from `i = 6` through `i = 1`, compares values to either side of `i` and records in `w` the rightmost value if it is less than the leftmost of the pair being compared. Values for `w` include `–1, 5, 2, 3`. Method returns the final value of `w = 3`.

11) A The `Scanner` is initialized with a `String` literal (`"seuss.txt"`) as the source rather than an actual `File` object. Parsing through the 9 characters in the `String`, `"seuss.txt"`, results in only a single line of text being read and added to the `List`. In order for the actual file contents to be parsed, the `Scanner` would have needed to be initialized as `new Scanner(new File("data.txt"))`.

12) B The contents of the `eight` array include `[0.0, 0.125, 0.25, 0.375, 0.5, 0.625, 0.75, 0.875]`.

13) D `((me – ((you / us) * me)) + (you * me)) = ((5 – ((24 / 3) * 5)) + (24 * 5))`

14) A `public class Random extends Object implements Serializable { ... }`

15) C `all = [`*reference to some*`]; some = [`*reference to all*`, 0, 2]`

16) A Default value for `boolean` primitive is `false`. Default value for `Boolean` object is `null`.

17) D `max = 2147483647; min = –2147483648`. Due to overflow, `–min = –2147483648` as well.

18) C Answer choice A produces random integers from the set of `[0, 1, 2]`. Answer choices B and D produce random integers from the set of `[–1, 0]`.

19) A `morse = ".-...---...----."`. String is split using the regular expression, `"."`, which splits on *any* single character (`"."` = regex wildcard pattern character) and would produce an array of 15 empty strings, However, `split()` does not include trailing empty strings in the resulting array, leaving `dashes` as an empty array.

20) E Sequential search finds index of last occurrence of `item` in `data`.

21) B Strings are removed from the `PriorityQueue` in ascending, lexicographic order. `'2' = 50, 'F' = 70, 'T' = 84, 'f' = 102`.

22) E After bitshifting 31 by 2 places to the left, `scan` $= 01111100_2 = 124_{10}$.

23) D $52_{15} = 77_{10}$

24) A Sums up the total number of positions each items are shifted during each insertion. The `'E'` is shifted by 0 positions. `'G'` is shifted by 1 position. `'I'` is shifted by 2 positions. `'O'` and `'N'` are each shifted by 3 positions. `0 + 1 + 2 + 3 + 3 = 9`.

25) E The resulting array is sorted in descending Unicode order.

26) C Removes and inserts each item in the list into descending order.

27) D Insertion sort yields $O(N^2)$ performance in the average and worst cases

28) B The regular expression requires the string to end with a non-digit character (e.g., `[^0–9]`).

29) B Returns a post-order traversal of the heap rooted at index `n`, where index [1] of `String src` represents the root of the overall heap.

30) E Throws a `StringIndexOutOfBoundsException` when `n = src.length()`.

31) B Any value XOR'ed with its bitwise complement will result in a value containing all 1 bits (i.e., 2's complement representation of –1).

32) A `base` references the head (node `[1]`) of a bi-directional, linked list consisting of `Disc` objects numbered as follows:
{WEST} `null ← [1] ↔ [7] ↔ [6] ↔ [5] ↔ [4] ↔ [3] ↔ [2] → null` *{EAST}*

33) D `base` references the head (node `[1]`) of a bi-directional, linked list consisting of `Disc` objects numbered as follows:
{WEST} `null ← [1] ↔ [7] ↔ [6] ↔ [3] ↔ [2] → null` *{EAST}*

34) C Since the initial `state` of node `[3]` is `false`, its `state` and the `state` of node `[6]` (to the `west` of node `[3]`) are each inverted to `true` (producing an output of "H" for each of those 2 nodes).

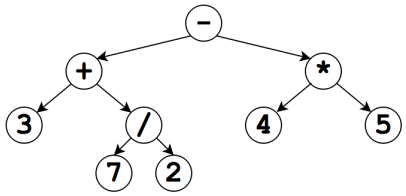
35) E Traversing `east` from node `[4]` leads to node `[3]` and node `[2]`, but traversing `west` from node `[2]` leads to node `[3]`, node `[6]`, node `[7]`, and node `[1]`.

36) B Each `Disc` object is a node in a bi-directional, linked list of `Disc` objects.

37) C Paths: `AGD`, `ABCGD`, `AFECGD`

★ANSWER KEY – CONFIDENTIAL★

38) A RPN is the post-order traversal of the following expression tree:



39) Any answer that equivalently expresses "(A Logical-AND C) Logical-OR (B Logical-AND D)" is acceptable (use of parentheses for correctly enforcing order of operations is *optional*):

```
(AC) + (BD)
(A * C) + (B * D)
(A && C) || (B && D)
(A and C) or (B and D)

(CA) + (BD)
(C * A) + (B * D)
(C && A) || (B && D)
(C and A) or (B and D)

(AC) + (DB)
(A * C) + (D * B)
(A && C) || (D && B)
(A and C) or (D and B)

(CA) + (DB)
(C * A) + (D * B)
(C && A) || (D && B)
(C and A) or (D and B)
```

```
(BD) + (AC)
(B * D) + (A * C)
(B && D) || (A && C)
(B and D) or (A and C)

(DB) + (AC)
(D * B) + (A * C)
(D && B) || (A && C)
(D and B) or (A and C)

(BD) + (CA)
(B * D) + (C * A)
(B && D) || (C && A)
(B and D) or (C and A)

(DB) + (CA)
(D * B) + (C * A)
(D && B) || (C && A)
(D and B) or (C and A)
```

40) $11011101_2 = -35$; $01011101_2 = 93$; $-35_{10} + 93_{10} = 58_{10}$

Conference _____ Code Number _____

UIL COMPUTER SCIENCE WRITTEN TEST

Questions (+6 points for each correct answer, -2 points for each incorrect answer)

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

FOR ADMINISTRATIVE USE ONLY

# Right:	×	6 pts	=	
# Wrong:	×	-2 pts	=	
# Skipped:	×	0 pts	=	0

	Score	Initials
Judge #1:		
Judge #2:		
Judge #3:		