University Interscholastic League
Computer Science Competition

Number 145 (District 1 - 2014)

General Directions:

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

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

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

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

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

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

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

8) All questions have ONE and only ONE correct (BEST) answer. There is a penalty for all incorrect answers.

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

Scoring:

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

Note: Correct responses are based on Java, **J2sdk v 1.7.25**, from Sun Microsystems, Inc. All provided code segments are intended to be syntactically correct, unless otherwise stated (i. e. `error` is an answer choice) and any necessary Java 2 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…** ***import static java.lang.System.*;***

QUESTION 1

Which of these is NOT equivalent to $10101_2 + 10000_2$?

A. 35_{10} B. 45_8 C. 25_{16} D. 100101_2 E. All are equivalent

QUESTION 2

For which initial values of p and q will the code on the right output `true`?

A. `p=true, q=true;` B. `p=false, q=true;`

C. `p=true, q=false;` D. `p=false, q=false;`

E. None of these

```
boolean p=<value1>, q=<value2>;
out.println(p&&q);
```

QUESTION 3

What is output by the code to the right?

A. `4` B. `4.0` C. `5`

D. `5.0` E. `6`

```
double a = 4.1573;
out.println(Math.ceil(a));
```

QUESTION 4

What is output by the code to the right?

A. `13.9` B. `15.7`

C. `27.0` D. `27.4`

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

```
double x = 13.7;
x = 2 * x;
out.println(x);
```

QUESTION 5

What is output by the code to the right?

A. `biminitop biminitop`

B. `biminitop bikinitop`

C. `bikinitop bikinitop`

D. `bikinitop biminitop`

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

```
String s = "biminitop";
String t = s.replace('m','k');
out.println(s+" "+t);
```

QUESTION 6

What is output by the code to the right?

A. `-4.0` B. `-5.0`

C. `-8.2` D. `-9.0`

E. `17.0`

```
out.printf("%.1f\n",9/2-6.5*2);
```

QUESTION 7

What is output by the code to the right?

A. `null` B. `null5` C. `5`

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

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

```
Integer x = null;
int y = 5;
out.println(x + y);
```

QUESTION 8

What is output by the code to the right?

A. `-50 -2 1`
B. `-49 -56 57`
C. `-51 -1 0`
D. `-51 -56 55`
E. `-50 -56 56`

```
int x = ~50;
int y = x/7<<3;
int z = ~y;
out.println(x+" "+y+" "+z);
```

QUESTION 9

What is output by the code to the right?

A. `Chill`
B. `Dude`
C. `Yo`
D. `Sup`
E. `DudeSupWordChill`

```
char a = 'e';
switch(a)
{
  case 'a':out.println("Yo");break;
  case 'e':out.println("Dude");break;
  case 'i':out.println("Sup");break;
  case 'o':out.println("Word");break;
  default :out.println("Chill");
}
```

QUESTION 10

What is output by the code to the right?

A. `7`
B. `6`
C. `5`
D. `4`
E. There is no output due to a compile error.

```
int x=0;
String [] a = {"red","white","blue"};
char[][]list=new char[a.length][];
for(String s:a)
  list[x]=a[x++].toCharArray();
int k=0;
for(char[]j:list)
  for(char m:j)
     k+="yellow".indexOf(m)>=0?0:1;
  out.println(k);
```

QUESTION 11

The `toString` method is partially implemented in the code to the right. Which statement below would **best** replace <statement1> so that the output in the client code shows `"6 string acoustic"`?

A. `return "6 string acoustic"`
B. `return numStrings + " string " + type`
C. `out.println("6 string acoustic")`
D. `out.println(numStrings + " string " + type)`
E. `"6 string acoustic"`

QUESTION 12

In what Java class is the `toString` method originally defined ?

A. `Guitar`
B. `Object`
C. `System`
D. `String`
E. `Scanner`

QUESTION 13

What term refers to redefining the `toString` method as shown in the code to the right ?

A. inheritance
B. overloading
C. overriding
D. polymorphism
E. interfacing

```
class Guitar
{
  private String type;
  private int numStrings;
  public Guitar()
  {
     type = "acoustic";
     numStrings = 6;
  }
  public Guitar(int n)
  {
     this();
     numStrings = n;
  }
  public Guitar(int n, String s)
  {
     this(n);
     type = s;
  }
  public String toString()
  {
     <statement1>;
  }
/////////////////////////////////
////client code
Guitar g = new Guitar();
out.println(g);
```

QUESTION 14

What is output by the code to the right?

A. `523` B. `637`

C. `790` D. `951`

E. `1003`

```
static int stuf(int [] list){
  int k=0,m=0;
  for(int x:list){
     int c=0;
     String s = Integer.toString(x);
     char []ss=s.toCharArray();
     for(char a:ss)
        c+=a-48;
     if(c>k){
        k=c;m=x;
     }
  }
  return m;
}
//client code
int [] list = {523,637,951,790,1003};
out.println(stuf(list));
```

QUESTION 15

What is output by the code to the right?

A. `630` B. `963` C. `9630`

D. There is no error, but there is no output

E. There is no output due to a compile error

```
for(int x=9; x==0;x-=3)
  out.print(x);
```

QUESTION 16

What is output by the code to the right?

A. `123456` B. `125456` C. `156456`

D. `153456` E. `433456`

```
int [] list = {1,2,3,4,5,6};
list[list[1]]=list[list[4]];
list[1]=list[list[3]];
for(int x:list)
 out.print(x);
```

QUESTION 17

What is output by the client code to the right?

A. `2.0` B. `6.0` C. `8.0`

D. `10.0` E. `14.0`

QUESTION 18

What term best describes the function of the `myst` method defined on the right?

A. Euclid's greatest common factor algorithm

B. Pascal's triangle

C. Leibniz integral rule

D. Newton's law of gravitation

E. Pythagorean theorem

```
public static double myst(double A,
double B)
{
  double AA = Math.pow(A,2);
  double BB = Math.pow(B,2);
  double C = Math.sqrt(AA+BB);
  return C;
}
//client code
double a = 6.0;
double b = 8.0;
out.println(myst(a,b));
```

QUESTION 19

What is output by the code to the right?

A. `1` B. `-1` C. `9` D. `-9` E. `0`

```
String a = "Auburn";
String b = "Alabama";
out.println(a.compareTo(b));
```

QUESTION 20

What is output by the code to the right?

A. `B4 114 1110010` B. `B4 176 10110000`

C. `B4 180 10110100` D. `B4 B416 1011010000010110`

```
String s = "B4";
int i = Integer.parseInt(s,16);
String t = Integer.toBinaryString(i);
out.println(s+" "+i+" "+t);
```

QUESTION 21

What is output by the code to the right?

A. `-21` B. `-27` C. `21`
D. `24` E. `27`

```
static int t(int x)
{
  return x%7>3?x-3:x+3;
}
//client code
out.println(t(24));
```

QUESTION 22

What is output by the code to the right?

A. `01100100` B. `0110010` C. `100`
D. `1100100` E. `1101100`

```
String s;
s = Integer.toBinaryString(100>>32);
out.println(s);
```

QUESTION 23

What is output by the code to the right?

A. `-1.00` B. `0.50` C. `0.71`
D. `1.00` E. `1.73`

```
double d = Math.log(Math.E);
out.printf("%.2f\n",d);
```

QUESTION 24

What is output by the code to the right?

A. `10000000000000000000000000000000` (1, 31 0s)
B. `100000000000000000000000000000000` (1, 32 0s)
C. `11111111111111111111111111111111` (32 1s)
D. `111111111111111111111111111111110` (32 1s, 0)
E. `10000000000000000000000000000001` (1, 30 0s, 1)

```
int x = Integer.MIN_VALUE;
String s = Integer.toBinaryString(x);
out.println(s);
```

QUESTION 25

Find f(-4) according to the recursive function definition shown on the right. You may use the space below to do your work.

f(-4) =

A. `-6` B. `-2` C. `3`
D. `5` E. `9`

```
         ┌ 2(f(x+2))-f(x+1)+1 when x<0
f(x,y)= ┤ 1                  when x=0
         └ 0                  when x>0
```

QUESTION 26

What is output by the code to the right?

A. `[3, 5, 1, 7]` B. `[4, 2, 6, 0]`
C. `[1, 3, 5, 7]` D. `[0, 2, 4, 6]`
E. There is no output due to a compile error.

```
int [] list = {3,4,2,5,1,6,7,0};
ArrayList<Integer> List1 = new
  ArrayList<Integer>();
ArrayList<Integer> List2 = new
  ArrayList<Integer>();
for(int x:list){
      if(x%2==0)
         List1.add(x);
      List2.add(x);
   }
List2.removeAll(List1);
out.println(List1);
```

QUESTION 27

What is output by the code to the right?

A. `I` B. `L`

C. `R` D. `S`

E. `T`

```
String s = "UILDISTRICTCONTEST";
char[]list = s.toCharArray();
int x=1;
PriorityQueue<Character> pq;
pq = new PriorityQueue<Character>();
for(char a:list){
     pq.offer(a);
     if(x%3==0){
        pq.poll();pq.poll();
     }
     x++;
  }
out.println(pq.peek());
```

QUESTION 28

What is output by the code to the right?

A. `5 4.0` B. `4 5.0`

C. `5.0 4.0` D. `4 5`

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

```
int j = 100;
double k = 20;
j/=k;
k/=j;
out.println(j+" "+k);
```

QUESTION 29

What is output by the code to the right?

A. `000 011 101 111` B. `000 011 100 110`

C. `001 011 101 111` D. `000 010 100 111`

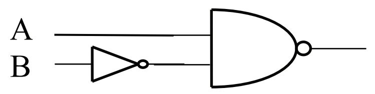
```
for(int p = 0; p <= 1; p++)
  for(int q = 0;q <= 1; q++)
     out.print(""+p+q+(p|q^p)+" ");
```

QUESTION 30

Which of the following logical statements is represented by the digital electronics diagram on the right ?

A. !A || !B B. !(A || !B)

C. !(A && !B) D. !A && !B



QUESTION 31

There is possibly something wrong with the code on the right that would cause a compile error, or it could be just fine. Which answer choice best describes the situation ?

A. There is nothing wrong…the code is fine as is.

B. The interface methods should not have semicolons

C. The class B method A1 needs something inside the {}

D. {} brackets are missing in the interface methods

E. The word `public` needs to precede each method definition.

QUESTION 32

Assuming the code is correct as is, or that the proper fix has been applied so that method A1 outputs the phrase “Hello World” and method A2 returns the value 0, what is the output of the client code listed?

A. `0` B. `HelloWorld0`

C. `HelloWorld`

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

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

```
interface A
{
  void A1();
  int A2();
}
class B implements A
{
  void A1(){}
  int A2(){return 0;}
}
//client code
A b = new B();
b.A1();
out.print(b.A2());
```

QUESTION 33

What is output by the code to the right?

A. `{a=5, b=7, e=3, f=7}`

B. `{a=5, b=7, c=3, e=3, f=7}`

C. `{c=4, e=3, b=7, a=5, f=7}`

D. `{e=3, b=7, a=5, f=7}`

E. `{a=5, e=3, f=7}`

```
Map<Character,Integer> m = new
  TreeMap<Character,Integer>();
m.put('c',4);
m.put('e',3);
m.put('b',7);
m.put('a',5);
m.put('c',3);
m.put('f',7);
m.remove('c');
out.println(m);
```

QUESTION 34

Which of these is the most efficient O(N) rating?

A. O(N) B. O(N^2) C. O(log N) D. O(N log N) E. O(1)

QUESTION 35

In the code to the right, what value is the last one popped ?

A. 3
B. 5
C. 6
D. 7
E. 9

```
Stack<Integer> s = new
   Stack<Integer>();
s.push(3);
s.push(5);
s.push(9);
s.pop();
s.push(6);
s.pop();
s.pop();
s.push(2);
s.push(7);
```

QUESTION 36

If A and B are Boolean values, which is the most simplified expression for A*0 + B + 1, where * means AND, + means OR, 0 means false, and 1 means true?

A. `0` B. `1` C. `A` D. `B` E. `A+B`

QUESTION 37

What is the length of the longest diagonal of 1s printed by this code?

A. 3
B. 4
C. 5
D. 7
E. 6

```
for(int x=0;x<8;x++) {
  for(int y=0;y<8;y++)
     out.print(((x+y)%4==0)?1:0);
     out.println();
  }
```

QUESTION 38

What is output by the code to the right?

A. `9` B. `10` C. `11`

D. `12` E. `16`

```
int a = 45;
int b = 34;
out.println(a%10+b/10+b%10);
```

QUESTION 39

In graph 1 on the right, the adjacency matrix would look like this, where 1 means a one way connection and 0 would mean no connection:

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

Which choice below represents the adjacency matrix for Graph 2 on the right?

A.

	A	B	C	D
A	0	1	1	0
B	1	0	0	1
C	0	0	1	0
D	0	1	0	0

B.

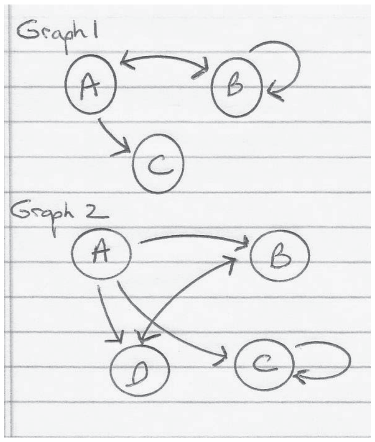
	A	B	C	D
A	0	1	1	1
B	0	0	0	1
C	0	0	0	0
D	0	1	0	0

C.

	A	B	C	D
A	0	1	1	1
B	0	0	0	1
C	0	0	1	0
D	0	1	0	0

D.

	A	B	C	D
A	0	0	0	0
B	1	0	0	1
C	1	0	1	0
D	1	1	0	0



QUESTION 40

What is output by the code to the right?

A. `16.0 15.0` B. `16.0 16.0`

C. `14.0 17.0` D. `12.0 18.0`

E. `5.0 20.0`

```
double a = 5,b=20;
do{
  if (a<b)
     a=a+(int)(b/a)+1;
  b=b-1;
  }
while(a<=b);
out.println(a+" "+b);
```

Standard Classes and Interfaces — Supplemental Reference

class java.lang.Object

- `boolean equals(Object other)`
- `String toString()`
- `int hashCode()`

interface java.lang.Comparable<T>

- `int compareTo(T other)`
  Return value < 0 if `this` is less than `other`.
  Return value = 0 if `this` is equal to `other`.
  Return value > 0 if `this` is greater than `other`.

class java.lang.Integer implements Comparable<Integer>

- `Integer(int value)`
- `int intValue()`
- `boolean equals(Object obj)`
- `String toString()`
- `int compareTo(Integer anotherInteger)`
- `static int parseInt(String s)`
- `static int parseInt(String s, int radix)`

class java.lang.Double implements Comparable<Double>

- `Double(double value)`
- `double doubleValue()`
- `boolean equals(Object obj)`
- `String toString()`
- `int compareTo(Double anotherDouble)`
- `static double parseDouble(String s)`

class java.lang.String implements Comparable<String>

- `int compareTo(String anotherString)`
- `boolean equals(Object obj)`
- `int length()`
- `String substring(int begin, int end)`
  Returns the substring starting at index `begin` and ending at index `(end - 1)`.
- `String substring(int begin)`
  Returns `substring(from, length())`.
- `int indexOf(String str)`
  Returns the index within this string of the first occurrence of `str`. Returns `-1` if `str` is not found.
- `int indexOf(String str, int fromIndex)`
  Returns the index within this string of the first occurrence of `str`, starting the search at the specified index.. Returns `-1` if `str` is not found.
- `charAt(int index)`
- `int indexOf(int ch)`
- `int indexOf(int ch, int fromIndex)`
- `String toLowerCase()`
- `String toUpperCase()`
- `String[] split(String regex)`
- `boolean matches(String regex)`

class java.lang.Character

- `static boolean isDigit(char ch)`
- `static boolean isLetter(char ch)`
- `static boolean isLetterOrDigit(char ch)`
- `static boolean isLowerCase(char ch)`
- `static boolean isUpperCase(char ch)`
- `static char toUpperCase(char ch)`
- `static char toLowerCase(char ch)`

class java.lang.Math

- `static int abs(int a)`
- `static double abs(double a)`
- `static double pow(double base, double exponent)`
- `static double sqrt(double a)`
- `static double ceil(double a)`
- `static double floor(double a)`
- `static double min(double a, double b)`
- `static double max(double a, double b)`
- `static int min(int a, in b)`
- `static int max(int a, int b)`
- `static long round(double a)`
- `static double random()`
  Returns a `double` value with a positive sign, greater than or equal to `0.0` and less than `1.0`.

interface java.util.List<E>

- `boolean add(E e)`
- `int size()`
- `Iterator<E> iterator()`
- `ListIterator<E> listIterator()`
- `E get(int index)`
- `E set(int index, E e)`
  Replaces the element at `index` with the object `e`.
- `void add(int index, E e)`
  Inserts the object `e` at position `index`, sliding elements at position `index` and higher to the right (adds 1 to their indices) and adjusts `size`.
- `E remove(int index)`
  Removes element from position `index`, sliding elements at position `(index + 1)` and higher to the left (subtracts 1 from their indices) and adjusts `size`.

class java.util.ArrayList<E> implements List<E>

class java.util.LinkedList<E> implements List<E>, Queue<E>

Methods in addition to the `List` methods:

- `void addFirst(E e)`
- `void addLast(E e)`
- `E getFirst()`
- `E getLast()`
- `E removeFirst()`
- `E removeLast()`

class java.util.Stack<E>
- `boolean isEmpty()`
- `E peek()`
- `E pop()`
- `E push(E item)`

interface java.util.Queue<E>
- `boolean add(E e)`
- `boolean isEmpty()`
- `E peek()`
- `E remove()`

class java.util.PriorityQueue<E>
- `boolean add(E e)`
- `boolean isEmpty()`
- `E peek()`
- `E remove()`

interface java.util.Set<E>
- `boolean add(E e)`
- `boolean contains(Object obj)`
- `boolean remove(Object obj)`
- `int size()`
- `Iterator<E> iterator()`
- `boolean addAll(Collection<? extends E> c)`
- `boolean removeAll(Collection<?> c)`
- `boolean retainAll(Collection<?> c)`

class java.util.HashSet<E> implements Set<E>

class java.util.TreeSet<E> implements Set<E>

interface java.util.Map<K,V>
- `Object put(K key, V value)`
- `V get(Object key)`
- `boolean containsKey(Object key)`
- `int size()`
- `Set<K> keySet()`
- `Set<Map.Entry<K, V>> entrySet()`

class java.util.HashMap<K,V> implements Map<K,V>

class java.util.TreeMap<K,V> implements Map<K,V>

interface java.util.Map.Entry<K,V>
- `K getKey()`
- `V getValue()`
- `V setValue(V value)`

interface java.util.Iterator<E>
- `boolean hasNext()`
- `E next()`
- `void remove()`

interface java.util.ListIterator<E> extends java.util.Iterator<E>

Methods in addition to the `Iterator` methods:
- `void add(E e)`
- `void set(E e)`

class java.lang.Exception
- `Exception()`
- `Exception(String message)`

class java.util.Scanner
- `Scanner(InputStream source)`
- `boolean hasNext()`
- `boolean hasNextInt()`
- `boolean hasNextDouble()`
- `String next()`
- `int nextInt()`
- `double nextDouble()`
- `String nextLine()`
- `Scanner useDelimiter(String pattern)`

Computer Science Answer Key
UIL District 1 2014

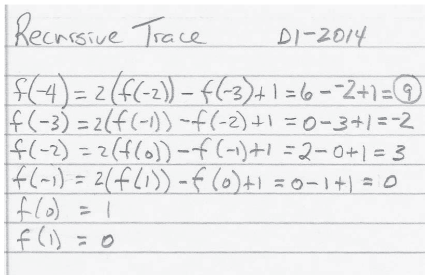
1)	A	11)	B	21)	E	31)	E
2)	A	12)	B	22)	D	32)	B
3)	D	13)	C	23)	D	33)	A
4)	D	14)	B	24)	A	34)	E
5)	B	15)	D	25)	E	35)	B
6)	D	16)	C	26)	B	36)	B
7)	E	17)	D	27)	D	37)	D
8)	D	18)	E	28)	A	38)	D
9)	B	19)	C	29)	A	39)	C
10)	A	20)	C	30)	C	40)	A

Note to Graders:

- All provided code segments are intended to be syntactically correct, unless otherwise stated (e.g. error is an answer). **Ignore any typographical errors.**
- Any necessary Standard Java 2 Packages are assumed to have been imported as needed.
- Assume any undefined (undeclared) variables have been defined as used.

Brief Explanations:

1. $10101_2 + 10000_2 = 21_{10} + 16_{10} = 37_{10} = 45_8 = 25_{16} = 100101_2$
2. For Boolean AND to be true, both inputs need to be true
3. The Math.ceil method returns the “rounded up” decimal value, in this case, 4.1573 goes up to 5.0
4. 13.7 times 2 is 27.4
5. The ‘m’ is replaced with ‘k’, making the new String “bikinitop”
6. 9/2 is 4, 6.5 * 2 is 13.0, and 4 – 13.0 is 9.0.
7. This is a runtime error (null pointer exception) since null cannot be added to an integer.
8. ~ means complement, or simply put, opposite, minus 1. ~50 is -51. -51/7 is -7, which is then multiplied by 8 (<<3) making -56. ~(-56) becomes 55.
9. The matching case for ‘e’ outputs the word “Dude”, and stops at the break
10. This code counts all the letters in “red”, “white”, and “blue” that are NOT in “yellow”, which are “rdhitbu”
11. The job of the toString method is to include all of the instance values of the object in some form, so the return statement that does that is the best answer, even though **return “6 string acoustic”** will do the job for this particular object, but not for an object with different values.
12. The Object class is the origin of the toString method
13. Overriding is the process of redefining a method inherited from a parent class. Overloading is when you have several methods in the same class with the same name, but different parameter signatures.
14. This method simply adds up the digits in each number. 637 and 790 both have the greatest sum, but since 637 came first, it is the answer.
15. This loop never happens since x==0 evaluates to false at the beginning, so there is no output.
16. List position 2 gets the value 6 (the element in position 5), and list position 1 gets the value 5 (the element in position 4).
17. This method simply calculates and returns the 3rd side of a right triangle...
18. …which is the Pythagorean theorem.
19. The first ***different*** letters in these two strings are ‘u’ and ‘l’, and ‘u’ has an ASCII value 9 greater than ‘l’.
20. The hex value B4 is simply 11(B) times 16, or 176, plus 4, or 180, which has a binary value of 10110100.
21. Since 24 mod 7 is 3, the ternary operator evaluates to false, and 24+3 is the result.
22. Any integer right shifted 32 positions is back to where it started, actually a right circle 32 to be precise. The binary value of 100 is 1100100.
23. The log of E is 1.0.
24. The minimum value for an int is -2147483648, which in binary is 1 with 31 zeros.
25. See the recursive trace on the right for the solution to this problem.
26. List1 only adds even numbers, while List2 adds all of them. The removeAll indeed removes all of the evens from List2, leaving the odds, but the output only asks for List1, which contains the evens.
27. This sequence effectively pushes three characters in priority order, then pops the front two, and repeats this process throughout the end of the string. In the first three, “UIL”, the “U” remains since “I” and “L” are alphabetically in front of the “U”, so they get popped.



Recursive Trace D1-2014

$f(-4) = 2(f(-2)) - f(-3) + 1 = 6 - {}^-2 + 1 = \boxed{9}$

$f(-3) = 2(f(-1)) - f(-2) + 1 = 0 - 3 + 1 = {}^-2$

$f(-2) = 2(f(0)) - f(-1) + 1 = 2 - 0 + 1 = 3$

$f(-1) = 2(f(1)) - f(0) + 1 = 0 - 1 + 1 = 0$

$f(0) = 1$

$f(1) = 0$

28. Even though j is an int, the /= shortcut has an automatic cast, so 100 divided by 20.0 still returns 5. k gets 20.0 / 5, which is 4.0.
29. The Boolean expression P OR Q XOR P simplifies to just P OR Q, which results in true for all combinations except for false false.
30. This is a simple Digital Electronics diagram, with A and NOT B going into a NOT AND gate, so the expression is NOT(A AND NOT B).
31. An interface requires all methods to be designated public for it to compile, so the fix is to put the word “public” before each method in both the interface and the class. {} is a sufficient implementation for the A1 void method, which is the way you would simply ignore a method you do not wish to implement with anything significant.
32. Given the description of what each method should do, the output here is obvious…”HelloWorld0”.
33. A TreeMap is similar to a mathematical function, in the fact that there can only be one mapping per key (for every x this is one and only one y). There can be, however, duplicate values, like the 7 mapped by both “b” and “f”. When the “c” is mapped again with the 3, the 4 is removed. But then the “c” mapping is removed altogether, so there is no “c” mapping at the output. Since it is a Tree mapping, the keys are in natural order.
34. The most efficient of all Big O classifications is O(1).
35. The sequence is this: push 3, push 5, push 9, pop 9, push 6, pop 6, pop 5, push 2, and push 7. The 5 was the last value popped.
36. A*0 is simply false, and goes away. B OR 1 simplifies to true since OR with true is always true, therefore the simplified expression here is just TRUE, or 1.
37. The diagonal spanning from row 7, col 1 up and to the left to row 1, col 7 has 7 1s in it, the longest in this matrix.
38. a%10 results in 5, b/10 is 3, and b%10 is 4. The sum 5+3+4 is 12.
39. An adjacency matrix is a classic way to express a graph situation. Study the example carefully and it will make sense.
40. The sequence of values through the loop execution are: 5.0 and 20.0 to start, then 12.0 and 18.0, 14.0 and 17.0, 16.0 and 16.0, and finally 16.0 15.0.