University Interscholastic League
Computer Science Competition

Number 146 (District 2 - 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 $100010_2 + 100000_2$?

A. 66_{10} B. 112_8 C. 42_{16} D. 1000010_2 E. All are equivalent

QUESTION 2

What is output by the code to the right?

A. `5 19.0 3.8` B. `5 19.0 3`
C. `5.0 19.0 3.8` D. `5 19 4`
E. There is no output due to a compile error.

```
int w = 5;
double z = 19;
double q = z/w;
out.println(w+" "+z+" "+q);
```

QUESTION 3

What is output by the code to the right?

A. `falsefalsefalse` B. `truefalsefalse`
C. `truefalsetrue` D. `truetruetrue`
E. `truetruefalse`

```
Integer x = 5;
Integer y = x;
out.print(x==y);
y = 5;
out.print(x==y);
y = new Integer(5);
out.println(x==y);
```

QUESTION 4

What is output by the code to the right?

A. `5` B. `6 7 8`
C. `5 6 7 8` D. `5 6 7`
E. There is no output.

```
int x = 5;
while (x<=7)
  out.print(x+++" ");
```

QUESTION 5

What is output by the code to the right?

A. `1` B. `2` C. `5`
D. `8` E. `9`

```
String s = "bassGuitar";
out.println(s.lastIndexOf("a"));
```

QUESTION 6

What is output by the code to the right?

A. `5` B. `6` C. `9`
D. `10` E. `11`

```
int list[] = {1,3,5,2,4};
out.println(list[1]+list[3]);
```

QUESTION 7

For which initial values of p and q will this expression output `false`?

A. `true true` B. `true false`
C. `false true` D. `false false`

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

QUESTION 8

What is output by the code to the right?

A. `00` B. `0`
C. `06` D. `66`
E. `60`

```
int z = 42;
if(z%7==0)
  out.print(z/7);
  out.println(z%7);
```

QUESTION 9

What is output by the code to the right?

A. `63 254` B. `254 63` C. `63 -2`
D. `508 31` E. `31 508`

```
int b = 127;
int c = 127;
out.println((b>>=2)+" "+(c<<=2));
```

QUESTION 10

What is output by the code to the right?

A. `-5.0` B. `-6.0`
C. `5.0` D. `6.0`
E. There is no output due to a compile error.

```
double f = -5.9423;
out.println(Math.floor(f));
```

QUESTION 11

Which statements would correctly replace <statement1> in the client code on the right to correctly modify the current Guitar object g into a 5 string bass guitar?

I. `g.getNumStrings(5);`
II. `g.setNumStrings(5);`
III. `g = new Guitar(5) ;`
IV. `g = new Guitar(5,"bass") ;`

A. I only
B. II only
C. III only
D. III and IV only
E. II, III, and IV only

QUESTION 12

Which statement would correctly replace `<statement2>` in the client code shown to output the type for the Guitar object g?

A. `out.println(g.getType());`
B. `out.println(g.setType("bass"));`
C. `out.println(g.getNumStrings());`
D. `out.println(g.setNumStrings(4));`
E. `out.println(g);`

QUESTION 13

Assuming the statements above have been correctly defined as described what is the output of the client code?

A. `4 string bass`
B. `5 string bass`
C. `bass: 4 string`
D. `bass: 5 string`
E. `6 string acoustic`

```
static 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 void setType(String s){
     type = s;
  }
  public String getType(){
     return type;
  }
  public void setNumStrings(int n){
     numStrings = n;
  }
  public int getNumStrings(){
     return numStrings;
  }
  public String toString()
  {
     return type+": "+numStrings+
     " string";
  }
/////////////////////////////////
////client code
Guitar g = new Guitar(4,"bass");
<statement1>
<statement2>
out.println(g);
```

QUESTION 14

What is output by the code to the right?

A. `5` B. `5.6`
C. `7` D. `7.3`
E. `9`

```
out.printf("%.1f\n",3*4.2-7);
```

QUESTION 15

What is output by the code to the right?

A. `abcdef` B. `defabc`

C. `cbafed` D. `fedcba`

E. There is no output due to a compile error

```
static void showGrid(char[][]g){
 for(int r=g.length-1;r>=0;r--){
  for(int c=g[0].length-1;c>=0;c--)
   out.print(g[r][c]);
 }
}
//client code
char[][]g={{'a','b','c'},
          {'d','e','f'}};
showGrid(g);
```

QUESTION 16

For which of these input values will the output be 9?

A. `240` B. `100` C. `600`

D. `250` E. `260`

```
 double d = <input>;
 int x=0;
 do{
    d/=2;
    x++;
 }while(d>=1.0);
 out.println(x);
```

QUESTION 17

What value is in position 4 after the client code to the right executes?

A. `6` B. `-1` C. `2`

D. `5` E. `4`

QUESTION 18

What is the greatest value in the list after the method call?

A. `0` B. `-1` C. `2`

D. `5` E. `4`

```
public static void Myst(int[]list){
  for(int j = 3;j<=5;j++)
     list[j]=list[j-2]-list[j-1];
}
//client code
int [] list = new int[6];
list[1]=5;
list[2]=2;
Myst(list);
```

QUESTION 19

Which of these choices could replace **`<statement1>`** to output the value 5?

I. substring(15)
II. substring(16)
III. substring(5,10)
IV. substring(7,12)
V. substring(10,16)

A. I only B. I, II, and III only

C. I, III, and IV only D. II, III, and IV only

E. All will work correctly to output the value 5

```
String a = "01234567890123456789";
out.println(a.<statement1>.length());
```

QUESTION 20

What is output by the code to the right?

A. `000 010 101 110` B. `000 010 101 111`

C. `001 011 101 110` D. `001 010 100 111`

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

QUESTION 21

What is output by the code to the right?

A. `2.0` B. `3.0` C. `4.0`

D. `5.0` E. `6.0`

```
double y = 42;
y %= 13;
y = ++y;
out.println(y);
```

QUESTION 22

What is output by the code to the right?

A. `1010` B. `1100` C. `10`

D. `00001010` E. `1110`

```
String s=Integer.toBinaryString(10);
out.println(s);
```

QUESTION 23

What is output by the code to the right?

A. `64` B. `8` C. `4`
D. `32` E. `16`

```
out.println(Short.SIZE);
```

QUESTION 24

What is output by the code to the right?

A. `-1.00` B. `0.00` C. `1.00`
D. `2.72` E. `3.14`

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

QUESTION 25

Find f(6,5) according to the recursive function definition shown on the right. You may use the space below to do your work.

f(6,5) =

A. `8` B. `6` C. `5`
D. `2` E. `3`

```
          ┌ 2+f(x-3,y-1)  when x>y and
          │                    x>0
f(x,y)= ─┤ 1+f(y,x)      when y>=x and
          │                    x>0
          └ 0             when x<=0
```

QUESTION 26

What is output by the code to the right?

A. `32` B. `31`
C. `1` D. `2`
E. There is no output due to a compile error.

```
int x = -1>>>32;
String s = Integer.toBinaryString(x);
out.println(s.length());
```

QUESTION 27

What is output by the code to the right?

A. `3a` B. `4a` C. `4ade`
D. `3vea` E. `4IL`

```
String s = "ILoveAParade";
String []a = s.split("[j-rM-Q]");
List<String> b = Arrays.asList(a);
out.println(b.size()+b.get(3));
```

QUESTION 28

What is output by the code to the right?

A. `34 114` B. `34 44`
C. `44 54` D. `35 45`
E. `34 54`

```
Integer i = 34;
String s = i.toString();
int x = i;
String t = Integer.toString (x,5);
out.println(s+" "+t);
```

QUESTION 29

What is output by the code to the right?

A. `winterwind winterwind`
B. `winterwind wimterwimd`
C. `suntersund sumtersumd`
D. `winterwind suntersund`
E. `winterwimd sumtersumd`

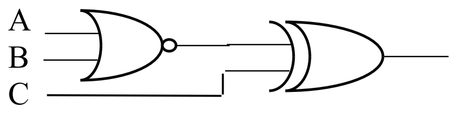
```
String w = "winterwind";
w.replaceAll("win","sun");
String s = w.replace('n','m');
out.println(w+" "+s);
```

QUESTION 30

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

A. A ^ B || C　　B. !(A || B) ^ C

C. !(A ^ B) || C　　D. A || B ^ C



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 abstract class methods should not have semicolons

C. The word `extends` should be `implements` instead

D. {} brackets are missing in the abstract class methods

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

QUESTION 32

Assuming the code is updated so that method A1 outputs the phrase `"I made a "` and method A2 returns the value `240`, what is the output of the client code listed?

A. `0`　　B. `240`

C. `I made a 240`

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

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

```
abstract class A{
  abstract void A1();
  abstract int A2();
}

class B extends 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. `3`　　B. `7`　　C. `9`

D. `5`　　E. `6`

```
Queue<Integer> q = new
        LinkedList<Integer>();
q.add(3);q.add(5);q.add(9);
q.poll();q.add(6);q.poll();
q.poll();q.add(2);q.add(7);
out.println(q.peek());
```

QUESTION 34

Which of these is the least 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

What is output by the code to the right?

A. `16`　　B. `15`　　C. 14

D. `13`　　E. `12`

```
String ss="Now is the time for all"+
  " good men to come to the aid of"+
  " their country";
String [] a = ss.split(" ");
Set<String> s = new
   HashSet<String>(Arrays.asList(a));
out.println(s.size());
```

QUESTION 36

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

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

QUESTION 37

What bottom-left-corner to top-right-corner diagonal series of characters is produced by this code??

A.	`abcde`	B.	`DEFGH`
C.	`ABCDE`	D.	`defgh`
E.	`01234`		

```
for(int x=0;x<5;x++){
 for(int y=0;y<5;y++)
  out.print(((x+y)%5==4)
        ?(char)(y+100):'-');
  out.println();
}
```

QUESTION 38

What is output by the code to the right?

A. `45657` B. `4565` C. `5657`

D. `7565` E. `5654`

```
LinkedList<Integer>a = new
LinkedList<Integer>();
a.push(4); a.add(5);
a.offer(6);a.add(3,5);
a.offerLast(7);a.pollFirst();
Iterator<Integer> i =
    a.descendingIterator();
while(i.hasNext())
         out.print(i.next());
```

QUESTION 39

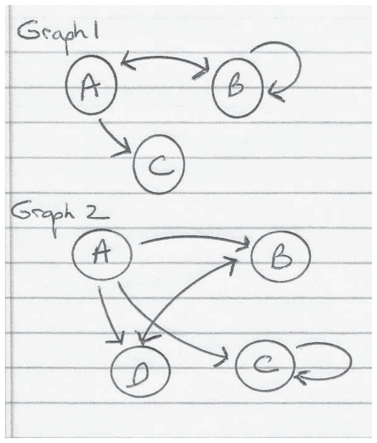
In graph 1 on the right, the adjacency matrix would look like this, where 1 means a one way connection and 0 would mean no connection:

$$\begin{array}{c|ccc} & A & B & C \\ \hline A & 0 & 1 & 1 \\ B & 1 & 1 & 0 \\ C & 0 & 0 & 0 \end{array}$$

How many zeroes would be in the adjacency matrix for Graph 2?

A. `6` B. `10` C. `16`

D. `13` E. `3`



QUESTION 40

What is output by the code to the right?

A. `10 10` B. `10 20`

C. `10 25` D. `25 10`

E. `20 20`

```
static void p(int []a,int []b){
  a[0]=a[0]+b[0];
  b[0]=a[0]-b[0];
  a=b;
}
//client code
int [] x={10};
int [] y={5};
p(x,y);
p(y,x);
out.println(x[0]+" "+y[0]);
```

No Test Material On This Page

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 2 2014

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

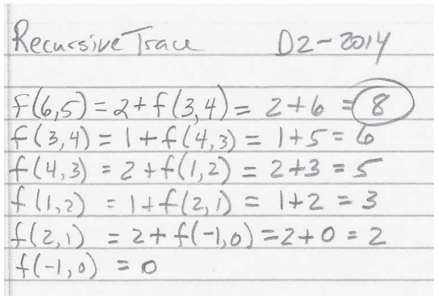
Note to Graders:

- 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.

Explanations:

1. $100010_2 + 100000_2 = 34_{10} + 32_{10} = 66_{10} = 102_8 = 42_{16} = 100010_2$
2. This is simple arithmetic. Just remember the data types for the output.
3. The first true result is obvious since both x and y reference the same object. For the **`y=5`** reassignment, there is a common memory section in Java for Strings and for smaller value integers that objects share when they are instantiated simply with the equals sign. Therefore, even though it looks like a separate object is created, it simply references the 5 that is in common memory, and therefore it is still pointing to the same memory location. However, when the **`new`** operator is used, a separate memory location is used, which results in **`false`** for the == operator.
4. Since the ++ is a post-increment operator, the value is output first, then the variable is incremented, with the result shown.
5. The **`lastIndexOf`** method is straight forward…the last index of the letter 'a' is in position 8 of the string.
6. Remembering that Java lists use zero based indexing (first element is in position zero), the elements in position 1 and 3 are 3 and 2, whose sum is 5.
7. The only way for the OR (||) operator to be false is when both Boolean values are false.
8. Both output statements are executed here, the first one because the ***`if`*** statement is true, and the second one regardless of the if statement since it is not attached to it, despite the indentation. The resulting output is simple math.
9. Right shift 2 is essentially dividing by 4 (2^2), and left shift 2 is multiplying by 4, with obvious results.
10. The floor function returns the nearest lower whole number value of the decimal, in this case, -6.0.
11. The traditional modifier method of classes starts with the word **`set`**, and in this case **`setNumStrings`** is the method to use, giving it the desired value as a parameter.
12. Similarly, the word **`get`** is the traditional prefix for accessor methods of instance variables, therefore **`getType`** is the one to use in this situation.
13. The toString method in this class definition lists the type first, followed by a colon, then the number of strings, and the word "string".
14. This is simple arithmetic. Nuff said.
15. This **`showGrid`** method outputs the entire matrix from bottom row to top, in right to left column order.
16. The value 260.0 divides into 130.0, 65.0, 32.5, 16.3, 8.13, 4.1, 2.03, 1.02, and finally 0.51, with 9 divisions. 250 requires only 8 divisions, and 600 requires 10.
17. The contents of the array at the start are: 0 5 2 0 0 0. After each loop iteration the contents are: 0 5 2 3 0 0, 0 5 2 3 -1 0, and 0 5 2 3 -1 4. Position 4 contains -1 at the conclusion of the method call.
18. The greatest value at the end is 5.
19. Since the length of the string is 20, the substring calls with 15, 5 and 10, and 7 and 12 all will return a string of length 5.
20. The expression **`p xor q and p`** simplifies to **`p and not q`**, which means the only true result is when p is true and q is false, indicated by 101 in the output. Using Boolean identities, the simplification sequence is as follows: **`p^q&&p = p&&!(q&&p) || !p&&q&&p = p&&(!q||!p) = p&&!q`**. You can also use the truth table process to evaluate this expression.
21. 42.0 % 13 results in 3.0, which is then incremented to become 4.0.
22. Decimal 10 in binary is 1010.
23. The short data type is stored in 16 bits of memory.
24. The natural log of E (`2.718281828459045`), the base of the natural logs, is 1.00.
25. The recursive trace for this question is shown on the right.
26. The binary representation for -1 is a string of 32 1s, which when right shifted 32 places circles back to the same 32 1s.
27. The split for this problem results in the following: `[IL, veA, a, ade]`, with a length of 4 and "`ade`" in position 3.
28. The base 5 equivalent of 34 is 114.
29. The replaceAll method does not change the existing String (Strings are immutable), but instead returns a new String with the modifications indicated. The original String w is not changed, however a new String **s** is created changing all 'n's to 'm's.
30. The A and B signals go into a NOR gate, which goes into an XOR gate with C, resulting in NOT(A OR B) XOR C.
31. This code is fine as is. Unlike the interface, the abstract class does not require the word `public` preceding the method name.
32. The call to methods A1 and A2 simply result in the output, "`I made a 240`".
33. The contents of the queue after each command are: [3], [3, 5], [3, 5, 9], [5, 9], [5, 9, 6], [9, 6], [6], [6, 2], [6, 2, 7], with 6 at the front.
34. The least efficient of these O(N) ratings is O(N^2), which is typically characterized by some nested loop process, such as an insertion sort or bubble sort.
35. Although there are 16 words in this sentence, only 14 are unique, which is what this code does (sets have no duplicates).
36. The expression A AND B AND A OR 0 simplifies to just A AND B, since the repeated A dissolves into just one A, and the OR 0 is the identity rule and effectively disappears.
37. Since 97 is the ASCII value for lower case 'a', 100 represents 'd', which is where this diagonal of characters starts, producing the series "defgh".
38. The contents of this list after each command is as follows: [], [4], [4, 5], [4, 5, 6], [4, 5, 6, 5], [4, 5, 6, 5, 7], [5, 6, 5, 7].
39. To find out the number of 1s in this matrix, simply count the number of arrows, which is 6. Since it is a 4X4 matrix, which means 16 elements, the remaining 10 elements are zeroes.
40. This one is tricky. The first two statements in the p method actually effect the actual parameters, the lists x and y, since arrays are passed by reference, but the third statement (a=b) does not. Even though **a** is reassigned to reference the **b** list in the method, this does not make the original x reference change, therefore it still points to its original list. Here is the state of each list after each command.

- x[0] = 10 y[0] = 5
- a[0] = 15 b[0] = 5
- a[0] = 10 b[0] = 10
- x[0] = 15 y[0] = 10
- a[0] = 25 b[0] = 15
- a[0] = 10 b[0] = 10
- x[0] = 10 y[0] = 25



Recursive Trace D2-2014

f(6,5) = 2+f(3,4) = 2+6 = $\boxed{8}$

f(3,4) = 1+f(4,3) = 1+5 = 6

f(4,3) = 2+f(1,2) = 2+3 = 5

f(1,2) = 1+f(2,1) = 1+2 = 3

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

f(-1,0) = 0