University Interscholastic League
Computer Science Competition

Number 150 (Invitational B - 2015)

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 $10111000_2 + AB_{16}$?

A. 163_{16} B. 355_{10} C. 543_8 D. 10110001_2 E. All are equivalent

QUESTION 2

What is the result of the expression shown?

A. 1 B. 0.56

C. 2 D. -1 E. 0

```
23 / 9 * 1 % 2 = _____
```

QUESTION 3

What is output by the code to the right?

A.
```
----*----*
    49.2
```
B.
```
----*----*
49.20
```
C.
```
----*----*
9.20
```
D.
```
----*----*
49.2
```
E.
```
----*----*
  49.20
```

```
System.out.println("----*----*");
System.out.printf("%4.2f",49.2);
```

QUESTION 4

What is output by the code to the right?

A. UILCOMPUTERSCIENCE2015

B. UOLCOMPUTORSCOONCO2015

C. OUOOLOCOOOMOPOUOTOOROSOCOONOCOO2O0O1O5O

D. UOLCOMPUTORSCONCO2015

E. UILCOMPUTERSCONCE2015

```
String s = "UILCOMPUTERSCIENCE2015";
s = s.replaceAll("IE","O");
out.println(s);
```

QUESTION 5

What is output by the code to the right?

A. false B. true

```
boolean p = true;
boolean q = false;
out.println(!(p&&q));
```

QUESTION 6

What is output by the code to the right?

A. 17.0 B. 11.5 C. 23.0 D. 12.7

E. There is no output due to an error.

```
int x = 8;
double y = 15;
out.println(Math.hypot(8,15));
```

QUESTION 7

What is output by the code to the right?

A. 11 B. 11.2 C. 8.0 D. 8

E. There is no output due to an error.

```
int i = 8;
double d = 1.4;
out.printf("%d",i*=d);
```

QUESTION 8

What is output by the code to the right?

A. 16 B. 160

C. 14 D. 40

E. 4

```
int x = 14,y=0;
switch(x%3)
{
 case 0:y+=0;break;
 case 1:y+=1;
 case 2:y+=2;break;
 case 3:y+=3;
 default:y*=10;
}
out.println(x+y);
```

QUESTION 9

What is output by the code to the right?

A. 78 B. 159 C. 128

D. 79 E. There is no output due to an error.

```
int x = 4;
for(;x<75;x++)
 x*=2;
out.println(x);
```

QUESTION 10

What is output by the code to the right?

A. 0 7 4 5 3 6 2 0 B. 0 6 3 4 2 5 1 0

C. 6 3 4 2 5 1 0 0 D. 7 4 5 3 6 2 0 0

E. There is no output due to an error.

```
int a=1;
int[]list1={5,3,1,2,4,0};
int[]list2=new int[8];
for(int x:list1)
 list2[x+1]=++a;
for(int x:list2)
  out.print(x+" ");
```

QUESTION 11

Below is a value in a data file called **"stuff.dat"**.

```
154
```

In the code segment to the right, which choice is best for **<statement 1>** in order to retrieve the data for output purposes?

A. `String n = f.nextLine();`

B. `int n = f.nextInt();`

C. `double n = f.nextDouble();`

D. `String n = f.next();`

E. All code segments will work properly

```
Scanner f = new Scanner(new
File("stuff.dat"));
<statement 1>
out.println(n);
```

QUESTION 12

What is output by the code to the right?

A. 8

B. 7

C. 6

D. 5

E. 4

```
double d = 100;
int x = 0;
while(d<1000)
{
 d*=1.5;
 x++;
}
out.println(x);
```

QUESTION 13

Here are three lines taken from the Java Order of Precedence chart. Which choice represents the correct order of precedence for these three lines?

I. `||`
II. `|`
III. *`expr++ expr--`*

A. III, I, II B. I, II, III C. III, II, I D. I, III, II E. II, I, III

QUESTION 14

The integer data type **byte** uses 8 bits of storage, which means it has 2^8, or 256 possible values. Which of the choices below indicates the range of values for this data type.

A. -128 to 128 B. -128 to 127 C. 1 to 256 D. 0 to 255 E. -127 to 128

QUESTION 15

The output for the code to the right is: `0 1 2 0 1 2 0 1 2`

Which choice replaces **<statement>** in the code to the right so that it compiles and runs correctly?

A. `double`
B. `Object`
C. `Integer`
D. `String`
E. `int`

```
Integer [] list={0,1,2};
ArrayList<Integer> aList = new
ArrayList<Integer>();
for(Integer x:list)
 aList.add(x);
<statement> [] list2 =
aList.toArray();
for(Integer x:list)
 out.print(x+" ");
for(Integer x:aList)
 out.print(x+" ");
for(<statement> x:list2)
 out.print(x+" ");
```

QUESTION 16

How many ordered triples make this boolean expression false?

$$\overline{A + B * C}$$

A. 2 B. 3 C. 4 D. 5 E. 6

QUESTION 17

What is output by the code to the right?

A. `-96` B. `78` C. `96` D. `-78`
E. There is no output due to an error.

```
char x = 'A';
char y = 'a';
int z = 3;
out.println((x-y)*z);
```

QUESTION 18

What is output by the code to the right?

A. `0 24 9 0 39`
B. `0 36 21 6 0`
C. `0 12 0 42 27`
D. `0 48 33 18 3`
E. `0 0 45 30 15`

```
int[][]nums = new int[5][5];
for(int x = 3;x<50;x+=3)
 nums[x%5][x%4+1]=x;
for(int x = 0;x<5;x++)
 out.print(nums[2][x]+" ");
out.println();
```

QUESTION 19

The two's complement system is all about representing negative numbers in binary. For example, the positive value 72 in 8-bit binary is **01001000**. To find the binary representation for -72 using two's complement, you use this easy conversion process. Start from the right and keep all zeroes the same until you reach the first 1 digit. Keep that 1 the same also, and flip everything else, with an 8-bit binary result of **10111000** for -72. With that in mind, which of the following choices represents the 8-bit binary representation of -118?

A. 10001100 B. 10001110 C. 10001010 D. 10001011 E. 10001101

QUESTION 20

What is the least restrictive running time for the worst case scenario for the quick sort algorithm?

A. O(N^2)

B. O(N)

C. O(1)

D. O(N log N)

E. O(log N)

QUESTION 21

What is output by the client code to the right?

A. 3 4 5 6 7 9 B. 5 7 3 9 4 6

C. 9 7 6 5 4 3 D. 6 4 9 3 7 5

E. Not possible to determine

QUESTION 22

Which of the six indicated <statements> in the code to the right need to be altered in order to reverse the sorting order?

I. <statement 1>
II. <statement 2>
III. <statement 3>
IV. <statement 4>
V. <statement 5>
VI. <statement 6>

A. II only

B. III and IV only

C. III only

D. IV, V, and VI only

E. I only

```
public static void quickSort (int a[], int
lo, int hi){
  <statement 1>
  if (lo >= hi) return;
  int left = lo;
  int right = hi;
  <statement 2>
  int pivot = a[(lo+hi)/2];

  while ( left < right)  {
    <statement 3>
    while (a[left] > pivot) left++;
    <statement 4>
    while (pivot > a[right]) right--;
    <statement 5>
    if (left <= right)    {
      swap (a, left, right);
      left++;
      right--;
    }
  }
  <statement 6>
  quickSort (a, lo, right);
  quickSort (a, left, hi);
}

public static void swap (int a[],
                         int i, int j){
  int tmp = a[i];
  a[i] = a[j];
  a[j] = tmp;
}

public static void outputList(int[]list)
{
 for(int x=0;x<list.length;x++)
    out.print(list[x]+" ");
 out.println();
}
//client code
 int[]list = {5,7,3,9,4,6};
 quickSort(list);
 outputList(list);
```

QUESTION 23

How many 'o's will be output in the code to the right?

A. 2 B. 3 C. 4 D. 5

E. There is no output due to an error.

```
String s =
"abcdefghijklmnopqrstuvwxyz";
char [] list = s.toCharArray();
char a = list[0];
while(a++!='z')
 for(char b:list)
    if("aeiou".indexOf(b,a-98)>=0)
       out.print(b);
```

QUESTION 24

What is output by the code to the right?

A. 1.7 B. 0.9 C. 0.5

D. 1.0 E. None of these

```
int angle = 45;
out.printf("%.1f\n",
    Math.tan(Math.toRadians(angle)));
```

QUESTION 25

What is output by the code to the right?

A. 01111101 B. 01111010 C. 11111101

D. 00000010 E. 11111110

```
byte c = -10;
c>>=2;
out.println(Integer.toBinaryString(c)
.substring(24));
```

QUESTION 26

On the right is a fairly common version of the binary search algorithm, a standard search process used in computer science. What required process (if any) needs to replace **//line A** in the client code for this algorithm to work properly?

A. Sort the list in ascending order.

B. Reverse the order of the list.

C. There is no required process. The list is fine as is.

D. Sort the list in descending order.

E. Process the list into a hash table.

QUESTION 27

What is output by the client code to the right?

A. 0 0

B. 7 -1

C. 7 -2

D. 0 -1

E. 8 -1

```
static int binarySearch(int[] elements,
                        int target) {
 int left = 0;
 int right = elements.length - 1;
 while (left <= right) {
  int middle = (left + right) / 2;
   if (target < elements[middle]){
      right = middle - 1;
   }
   else if (target > elements[middle]){
      left = middle + 1;
   }
   else
      return middle;
 }
 return -1;
}
<client code>
int [] list = {5,-7,3,9,4,8,-3, 1,-5, 0};
//line A
out.print(binarySearch(list,5)+" ");
out.println(binarySearch(list,-6));
```

QUESTION 28

Which statement best describes the string patterns listed below as each replaces the **<pattern>** segment in the code to the right?

I. ".*" II. ".+" III. ".?"

A. All produce true outputs

B. All produce false outputs

C. II and III produce true, I produces false

D. I and II produce true, III produces false

E. I and III produce true, II produces false

```
String s = "Invitational B";
boolean p = Pattern.matches(<pattern>,s);
out.println(p);
```

QUESTION 29

What is output by the code to the right?

A. 0 -2 3 B. 6 10 -5

C. 5 11 3 D. 5 -2 3

E. There is no output due to an error.

```
 public static int myst(int x)
 {
  if(x<0)return 3;
  if(x==0)return 1;
  return myst(x-3)+2;
 }
//client code
out.print(myst(6)+" ");
out.print(myst(10)+" ");
out.println(myst(-5));
```

QUESTION 30

What is the output of the code to the right?

A. `BBBBAAAABBB`

B. `AAAABBBBAAA`

C. `AAABBBBBBAA`

D. `BBBAAAAAABB`

E. There is no output due to an error.

```
String s = "UIL 2015 CS";
char[]list=s.toCharArray();
for(char a:list)
 out.print(a>60?"A":"B");
```

QUESTION 31

What is output by the code to the right?

A. `7 28 59`

B. `7 34 63`

C. `7 42 59`

D. `7 42 77`

E. None of these

```
out.print(Integer.toOctalString(7));
out.print(Integer.toOctalString(34));
out.println(Integer.toOctalString(63));
```

QUESTION 32

Using the space provided, create a binary search tree using the letters, INVITATIONALB. After creating the tree, select the choice that shows <u>how many parent nodes have only one child</u>, and the <u>height of the resulting tree.</u>.

Assume that the initial tree shown has a height of zero, and that duplicate letters **<u>are allowed in the tree</u>**, and slide to the **left** of matching elements.

I

/ \

A. 5 6

B. 6 5

C. 6 6

D. 5 5

E. None of these

QUESTION 33

What is output by the statements in **section 1** of the client code below?

```
//section 1
ThingOne t1 = new ThingOne();
out.print(t1);
t1.setThing(6);
out.print(t1);
//section 2
t1=new ThingTwo(3);
out.print(t1.getThing());
//section 3
t1.reduceThing(1);
out.println(t1);
```

A. F5F6

B. AFDF

C. AF5DF6

D. A5D6

E. None of these

QUESTION 34

What is output by the statements in **section 2** of the client code in question 33?

A. HAI3

B. AHI3

C. AHI5

D. HI3

E. HAI5

QUESTION 35

What is output by the statements in **section 3** of the client code in question 33?

A. KLF7

B. KL52

C. L52

D. KLF52

E. KL7

```
class ThingOne{
 public ThingOne(){
  thing = 5;
  out.print("A");
}
 public ThingOne(int t){
  thing = t;
  out.print("B");
}
 public int getThing(){
  out.print("C");
  return thing;
}
 public void setThing(int t){
  thing = t;
  out.print("D");
}
 public void reduceThing(int t){
  thing -=t;
  out.print("E");
}
 public String toString(){
  out.print("F");
  return ""+thing;
}
 private int thing;
}
class ThingTwo extends ThingOne{
 public ThingTwo(){
  super();
  out.print("G");
  thing = 4;
 }
 public ThingTwo(int t){
  thing = t;
  out.print("H");
}
 public int getThing(){
  out.print("I");
  return thing;
}
 public void setThing(int t){
  thing = t;
  out.print("J");
}
 public void reduceThing(int t){
  thing -=t;
  out.print("K");
}
 public String toString(){
  out.print("L");
  return super.toString()+thing;
}
 private int thing;
}
```

QUESTION 36

Infix notation is the kind normally used in algebraic expressions, such as **`3 + 5 * 6`**, where the operators are between the operands. However, there is also prefix notation, where the operators are before the operands, such as **`+ 3 * 5 6`**, and postfix notation, operators after operands, like this: **`3 5 6 * +`**. Notice carefully that the operands never move around: only the operators change places. Here is another example: the infix expression **`6 * 7 + 9 - 8 * 2`** translates the prefix expression **`- + * 6 7 9 * 8 2`**, and **`6 7 * 9 + 8 2 * -`** for postfix. Given these examples to examine and study carefully, which of the prefix expressions below matches the infix expression shown?

`A+B-C*D+E^F`

A. **`+AB+*CD-^EF`**

B. **`+-AB-*CD^EF`**

C. **`+-+AB*CD^EF`**

D. **`++-AB*CD^EF`**

E. None of these

QUESTION 37

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

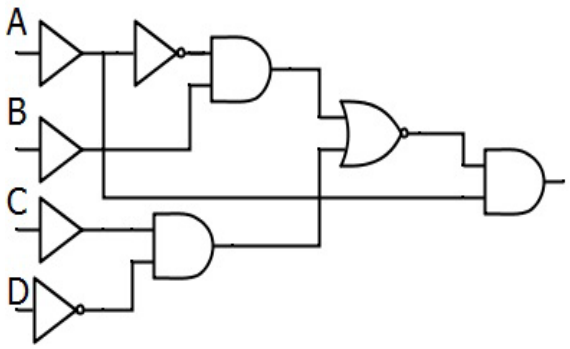
A. $\overline{(\overline{A}+B)*(C+\overline{D})}+A$

B. $(\overline{A}*B+C*\overline{D})*A$

C. $\overline{(\overline{A}*B+C*\overline{D})}*A$

D. $\overline{(\overline{A}*B+C*\overline{D})*A}$

E. None of these



QUESTION 38

Find f(-10) according to the recursive function definition shown below.

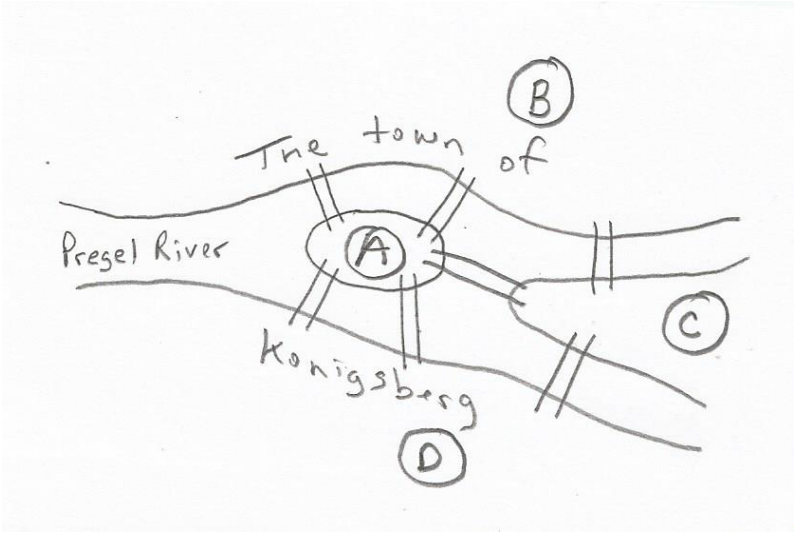
f(-10) =

```
f(x)=     f(x+5)+f(x+8)     when x<=0
f(x)=     2                 if x>0 and even
f(x)=     -1                if x>0 and odd
```

A. 0 B. 1 C. 2 D. 3 E. 4

QUESTION 39

A well-known graph problem studied by Leonard Euler had to do with the town of Konigsberg, in which there was an island at the point where the river Pregel forked. There were seven bridges connecting the island with the two banks of the river and the land between the forks, as shown in the picture. His question was whether or not there was a way to cross the seven bridges in a continuous walk through town without recrossing any of them. For example, he might start a path starting on the B side of town, cross over one of the two bridges to the island (A) and then on across to the south bank (D), and then across to the land area between the forks of the river (C). This path would be indicated by the letter sequence BADC. In this case study of the **Seven Bridges of Konigsberg**, a complete **Euler path**, or **Euler tour,** would be indicated by 8 letters. There could be any number of combinations.



Write the **sequence of eight letters** that represent your version of the Euler path for the seven bridges of Konigsberg. If you determine that it is not possible, write the answer "NOT POSSIBLE".

_ _ _ _ _ _ _ _

QUESTION 40

Simplify this expression to have only two operators and two NOTs. The allowable operators include AND(*), OR(+), XOR(⊕), and NOT (over bar).

$A * (\overline{B} + \overline{C}) + A * \overline{(B + C)}$

Computer Science Answer Key
UIL Invitational B 2015

1)	D	11)	E	21)	C	31)	D
2)	E	12)	C	22)	B	32)	B
3)	B	13)	C	23)	C	33)	C
4)	E	14)	B	24)	D	34)	B
5)	B	15)	B	25)	C	35)	D
6)	A	16)	D	26)	A	36)	C
7)	A	17)	A	27)	B	37)	C
8)	A	18)	C	28)	D	38)	B
9)	D	19)	C	29)	C	39)	NOT POSSIBLE
10)	A	20)	A	30)	C	40)	$A * (\overline{B} + \overline{C})$

Note: Since AND and OR have the commutative property, any answer that is a correctly commuted version of this answer is correct.

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:

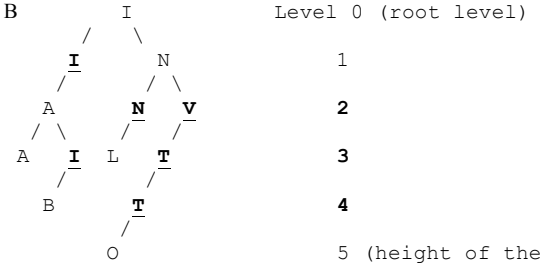
1. D $10111000_2 + AB_{16} = 184_{10} + 171_{10} = 355_{10} = 543_8 = 163_{16} = 101100011_2$
2. E 23 / 9 * 1 % 2 = 2 * 1 % 2 = 2 % 2 = 0
3. B Since a field width of 4 is not enough to contain 49.20, the compiler just takes what it needs and left justifies the output.
4. E The "IE" pattern finds the only "IE" in the word and replaces it with a single "O".
5. B Since p is true and q is false, p and q is false, which makes the NOT of P and Q true.
6. A This a lesser known Pythagorean triple...8, 15, 17. The Math.hypot method returns the hypotenuse for sides of 8 and 15, which when applied to the Pythagorean theorem becomes 64 + 225, or 289, whose square root is 17.
7. A 8 * 1.4 = 11.2, which truncates to 11 in the autocast provided by *=
8. A 14%3 equals 2, which results in a y value of 2 in the switch statement, and an output value of 14+2, or 16.
9. D The trace values of x are 4, 8, 9, 18, 19, 38, 39, 78, and 79, with the output value being 79.
10. A The sequence of the first loop is as follows: position 6 (5+1) of list2 gets the value ++a (2), 4 gets 3, 2 gets 4, 3 gets 5, 5 gets 6, and 1 gets 7. Positions 0 and 7 of list2 remain the default values 0. The second loop simply outputs the values of list2.
11. E For output purposes only, any of these statements will work with this data.
12. C The value sequence for d and x is: 100.0 0, 150.0 1, 225.0 2, 337.5 3, 506.25 4, 759.375 5, 1139.0625 6.
13. C The *expr++ expr--* (postfix) operators are on line 1 of the chart, followed by **|** on line 10, and **||** on line 12.
14. B Half of the values are negative (-128 to -1), and the other half non-negative (0-127).
15. B This toArray method of the ArrayList class returns an Object array, therefore the word Object must fill the <statement> in the code.
16. D

A	B	C	$\overline{A}$	$B * C$	$\overline{A} + B * C$	$\overline{\overline{A} + B * C}$
0	0	0	1	0	1	0
0	0	1	1	0	1	0
0	1	0	1	0	1	0
0	1	1	1	1	1	0
1	0	0	0	0	0	1
1	0	1	0	0	0	1
1	1	0	0	0	0	1
1	1	1	0	1	1	0

17. A 65 - 97 is equal to -32, which when tripled equals -96.
18. C The contents of the matrix after the assignment loop are:

```
0 0 45 30 15
0 36 21 6 0
0 12 0 42 27
0 48 33 18 3
0 24 9 0 39
```

19. C 118 in 8-bit binary is 01110110. Applying the rule mentioned in the problem gives you 10001010, which is the binary equivalent of -112.
20. A The worst case running time for a quick sort is O(N^2), which occurs when the list is already sorted, or when only a few elements are not in sorted order.
21. C This quick sort arranges a list of numbers in **descending** order due to the given comparison operators in statements 3 and 4. The output is the choice that shows all of the numbers in order from greatest to least.
22. B The comparison operators in both statements 3 and 4 need to be reversed for the quick sort to work in reverse order than is indicated here.
23. C The output string is "aeioueiouiououu", which contains four letter 'o's.
24. D The tangent of a 45 degree angle is 1.0 (opposite over adjacent)
25. C The value 10 in 8-bit binary is 00001010, which when made negative becomes 11110110 in twos complement form. When right shifted twice, it becomes 11111101, which is the decimal value -3.
26. A A precondition of this binary search algorithm is that the list is in ascending sorted order.
27. B The sorted list is -7, -5, -3, 0, 1, 3, 4, 5, 8, 9. Position 7 contains the 5, and -6 is not in the list, which is indicated by -1.
28. D The pattern ".*" matches zero or more of any character, ".+" matches one or more of any character, and the "?." matches one character, once or not at all.
29. C The recursive call value sequence for each initial call is: 6, 3, 0, result 2+2+1 = 5, 10, 7, 4, 1, -2, result 2+2+2+2+3 = 11, -5, result 3
30. C Since all alpha characters are above 60 in value, and all numeric and space characters are below, "A" is printed for the letters, and "B" is printed for the digits and the space character with this ternary operator statement.
31. D The base ten values of 7, 34, and 63 convert to base 8 values of 7, 42 (8 goes into 34 4 times with 2 remainder), and 77 (8 goes into 63 7 times with 7 remainder).
32. B



Level 0 (root level)

1

2

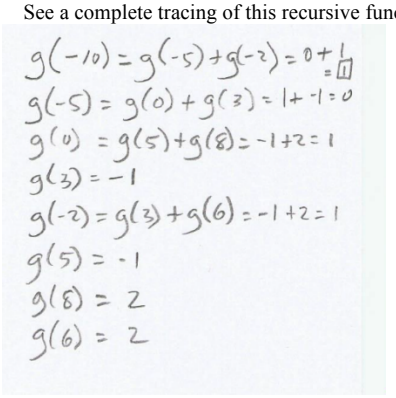
3

4

5 (height of the tree)

There are 6 parent nodes with only one child, shown in bold and underlined, and tree height is 4 (root node is at level zero, and furthest leaf is at level 4). In this binary search tree exercise, duplicates are allowed and are inserted to the left of matching elements. The first node is at level zero, therefore the resulting tree, as you can see in the key above, has a height of 4, with 5 leaves.

33. C The sequence of the first four statments is: call to default constructor(A), call to toString method(F) which returns and outputs the value 5, call to setThing(D) which changes the instance field value to 6, and a final call to the toString method(F), which returns and outputs the value 6.
34. B The sequence of the next two statments is: call to one parameter constructor(H) which assigns 3 to the ThingTwo instance field, but first automatically calls the ThingOne default constructor(A) which assigns 5 to the ThingOne instance field, followed by the call to the ThingTwo getThing method(I), which returns and outputs the instance field value of the ThingTwo object(3).
35. D The sequence of the last two statements is: call to reduceThing(K) which reduces the ThingTwo instance field value to 2, then call to the ThingTwo toString method(L), which in turn calls the ThingOne toString method(F), and returns concatenated values of both instance fields, 5 and 2.
36. C The first rule is that the operands stay in the same order, A B C D E F. Then follow order of operations. The exponent goes in front of the E and F, and the * before CD. The first + goes in front of the AB, then the minus before that, then finally the last + in front of everything.
37. C The signals NOT A and B go into an AND gate, which feeds into a NOT OR gate, which receives an AND signal from C and NOT D. The NOT OR result goes into an AND gate which also receives the A signal.
38. B See a complete tracing of this recursive function call shown above.



39. No matter where you start in this graph, it is **not possible** to cross all seven bridges without recrossing any of them. However, adding an 8th bridge, perhaps another between C and D, would enable an Euler path such as this: BADABCDCA. This problem helped give birth to the field of graph theory the development of which Euler is given much credit.

40. $A * (\overline{B} + \overline{C})$

Explanation: In the diagram below, the distributive property is used to simplify the original expression. Absorption is used by the second term to eliminate the third term, and then the A is factored out, resulting in the final expression, which contains two operators, AND and OR, with NOTs over both the B and C terms.