UIL Computer Science Written Test

2024 Invitational A

January/February 2024

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(begin, 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)
```

STANDARD CLASSES AND INTERFACES – SUPPLEMENTAL REFERENCE

Package java.util.function

```
Interface BiConsumer<T,U>
  void accept(T t, U u)

Interface BiFunction<T,U,R>
  R apply(T t, U u)

Interface BiPredicate<T,U>
  boolean test(T t, U u)

Interface Consumer<T>
  void accept(T t)

Interface Function<T,R>
  R apply(T t)

Interface Predicate<T>
  boolean test(T t)

Interface Supplier<T>
  T get()
```

UIL COMPUTER SCIENCE WRITTEN TEST – 2024 INVITATIONAL A

Note: Correct responses are based on **Java SE Development Kit 20 (JDK 20)** from Oracle, 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 20 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 numbers is not equal to the other four?

A) 1111010_2 **B)** 750_8 **C)** $7A_{16}$ **D)** 101_{11} **E)** 1322_4

Question 2

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

A) 4 **B)** 7 **C)** 8 **D)** 10 **E)** 12

```
out.print(22 / 7 + 15 % 8);
```

Question 3

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

A)
```
306
2
09
```

B)
```
2286
2
49
```

C)
```
306
2
49
```

D)
```
30
6
2
0
9
```

E)
```
36
2
49
```

```
out.print(22 + 8);
out.println(3 * 2);
out.println(7 / 3);
out.print(4 % 11);
out.print(12 - 3);
```

Question 4

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

A) A **B)** B **C)** C **D)** K **E)** P

```
String St = "BACKPACK";
int T = St.length()-1;
out.print(St.charAt(T-1));
```

Question 5

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

A) true

B) false

```
boolean A = true;
boolean B = false;
out.print(A && (true || B));
```

Question 6

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

A) 6 **B)** 6.0 **C)** 7 **D)** 7.0 **E)** 49.0

```
   double T = Math.sqrt(40);
out.print(Math.ceil(T));
```

Question 7

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

A) 25 **B)** 26.0 **C)** 26.5 **D)** 27.0 **E)** 27

```
double R = 7.5;
double S = 8.5;
double T = 21 / 2;
out.print(R + S + T);
```

Question 8

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

A) 5

B) 67

C) 7

D) 567

E) 6

```
int M = 5;
if(M > 10)
 {
  out.print(M);
  M++;
 }
if(M < 3)
  out.print(M);
  M++;
if(M == 7)
  out.print(M);
out.print(M);
```

Question 9

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

A) 6 11 16 21

B) 5 10 15 20 25

C) 51 101 151 201

D) 6 12 18 24

E) 6 11 16 21 26

```
int N = 5;
for(int x = N; x < N*N; x=x+N)
  out.print(x+1 + " ");
```

Question 10

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

A) 0 **B)** 1 **C)** 2 **D)** 3 **E)** 4

```
int[] food = {5,2,3,4,1,0};
out.print(food[food[0]-food[4]]);
```

Question 11

What is output by the code segment to the right?

A) 70

B) 68

C) 59

D) 45

E) 35

```
String St = "12 23 34 45 56 67";
Scanner Bob = new Scanner(St);
int M = Bob.nextInt();
Bob.next(); Bob.next(); Bob.next();
M = M + Bob.nextInt() + 2;
out.print(M);
```

Question 12

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

A) 3 6 9 12 15 18 21 24 27 30

B) 1 4 7 10 13 16 19 22 25 28

C) 3 6 9 12 15 18 21 24 27 30

D) 3 9 15 21 27

E) 6 12 18 24 30

```
for(int i = 1; i <= 30; i = i + 2)
  if (i%3==0)
    out.print(i+" ");
```

Question 13

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

A) 2048 **B)** 2044 **C)** 2040 **D)** 1024 **E)** 1022

```
int w = 511;
w = w<<3;
w = w>>2;
w = w++ + w++;
out.print(--w);
```

Question 14

What is the output of the code segment shown on the right?

A) 2 **B)** 4 **C)** 8 **D)** 16 **E)** 32

```
out.println(Double.SIZE/Byte.SIZE);
```

Question 15

What is output by the code segment to the right?

A) [36, 48, 60, 72, 84, 96]

B) [48, 60, 72, 84, 96]

C) [24, 36, 48, 60, 72, 84]

D) [0, 12, 24, 36, 48, 60, 72, 84, 96]

E) [96, 12, 24, 36, 48, 60, 72, 84, 96]

```
ArrayList<Integer> numbers;
numbers = new ArrayList<Integer>();
numbers.add(96);
for(int x=0; x<=100; x=x+12)
  numbers.add(x);
for(int x=1; x<=5; x=x+1)
  numbers.remove(0);
out.println(numbers);
```

Question 16

What is the output of the code segment shown on the right?

A) 3412 **B)** 34512 **C)** 46 **D)** 34523 **E)** 347

```
String yes = "012345678901234";
int P = yes.indexOf("23",5);
String no = yes.substring(3,5);
out.println(no + P);
```

Question 17

What is the output of the code segment shown on the right?

A) 180 **B)** 195 **C)** 196 **D)** 224 **E)** 225

```
int x,y;
for(x=1,y=20;x<y;x++,y--)
  x++;
out.println(x*y);
```

Question 18

What is the output of the code segment shown on the right?

A) 5 **B)** 7 **C)** 9 **D)** 11 **E)** 13

```
out.print((12 | 9) ^ (14 & 10));
```

Question 19

What is the output of the code segment shown on the right?

A) 81 **B)** 131 **C)** 212 **D)** 343 **E)** 555

```
int[]Special = new int[20];
Special[0] = 2;
Special[1] = 5;
for(int x=2; x<=19; x++)
  Special[x]=Special[x-2]+Special[x-1];
out.println(Special[10]);
```

Question 20

In the code to the right, what is output on line #1?

A) 1024 **B)** 512 **C)** 256 **D)** 128 **E)** 64

Question 21

In the code to the right, what is output on line #2?

A) 0 **B)** 1 **C)** 2 **D)** 3 **E)** 4

Question 22

In the code to the right, what is output on line #3?

A) 31 **B)** 32 **C)** 250 **D)** 255 **E)** 305

```
int[] Moe = new int[100];
int[] Larry = new int[100];
int[] Shemp = new int[100];
for(int x=0; x<Moe.length; x++)
  Moe[x] = x*x;
for(int x=0; x<Larry.length; x++)
  Larry[x] = Moe[x]%5;
for(int x=0; x<Shemp.length; x++)
  Shemp[x] = Moe[x] - Larry[x];
out.println(Moe[32]);   //line 1
out.println(Larry[11]); //line 2
out.println(Shemp[16]); //line 3
```

Question 23

What is the output of the code segment shown on the right?

A) 2 **B)** 8 **C)** 10 **D)** 12 **E)** 14

```
String ten = "JABAAAABBBBZAB";
for(int x=1; x<=100; x++)
  ten = ten.replaceAll("AB","");
out.print(ten.length());
// Note: There is no space between quotes on "" above
```

Question 24

What is the output of the code segment shown on the right?

A) 8 **B)** 9 **C)** 10 **D)** 11 **E)** 12

```
int T = 5;
for(int x = 0; x < T; x = x + 2)
  T++;
out.println(T);
```

Question 25

What is returned by the method call `Park(10,10)`

A) 3 **B)** 4 **C)** 5 **D)** 6 **E)** 7

Question 26

What is returned by the method call `Park(7,5)`

A) 13 **B)** 14 **C)** 15 **D)** 16 **E)** 19

Question 27

What is returned by the method call `Park(3,12)`

A) 44 **B)** 48 **C)** 52 **D)** 59 **E)** 102

```
public static int Park(int A, int B)
{
  if (A==B)
    return (A+B)/3;
  if (A > B)
    return Park(A-1, B) + A;
  return Park(A, B-2) + B;
}
```

Question 28

In the code to the right, what is output on line #1?

A) 12 **B)** 60 **C)** 64 **D)** 75 **E)** 90

Question 29

In the code to the right, what is output on line #2?

A) [15, 30, 45, 64, 90, 75, 60]

B) [75, 30, 12, 15, 90, 45, 64]

C) [15, 30, 64, 45, 75, 90, 60]

D) [15, 30, 45, 64, 75, 90, 60]

E) [15, 30, 45, 60, 64, 75, 90]

Question 30

IIn the code to the right, what is output on line #3?

A) [45, 60, 64, 75, 90]

B) [45, 90, 60, 64, 75]

C) [45, 60, 75, 64, 90]

D) [45, 64, 60, 90, 75]

E) [45, 60, 90, 64, 75]

```
PriorityQueue<Integer> low;
low = new PriorityQueue<Integer>();
low.add(60); low.add(75);
low.add(30); low.add(12);
low.add(15); low.add(90);
low.add(45); low.add(64);
out.println(low.remove());//line 1
out.println(low);         //line 2
low.remove();
low.remove();
out.println(low);         //line 3
```

Question 31

What is the output of the code segment shown on the right?

A) 15 **B)** 25 **C)** 35 **D)** 45 **E)** 55

```
int frog = 0;
for(int x=1; x<=10; x=x+1)
{
  frog--;
  for(int y=1; y<=x; y=y+1)
    frog++;
}
out.println(frog);
```

Question 32

In the client code to the right, what is output on line #1?

A) Bella

B) Dylan

C) BellaDylan

D) DylanBella

E) null

Question 33

In the client code to the right, what is output on line #2?

A) 33

B) 44

C) 55

D) 77

E) 88

Question 34

In the client code to the right, what is output on line #3?

A) 33

B) 44

C) 55

D) 77

E) 88

```
public class Mom
 {
  public String Name;
  public int ID;

  public Mom()
   {
     Name = "Bella";;
     ID = 33;
   }
}
  public class Son extends Mom
   {
     public String Name;

     public Son(String St, int R)
     {
       super();
       Name = St;
     }

    public Son()
     {
       super();
       ID  = 44;
     }
   }
//client code
Mom A = new Mom();
Son B = new Son("Dylan",55);
Son C = new Son();
out.println(A.Name);  // line 1
out.println(B.ID);    // line 2
out.println(C.ID);    // line 3
```

Question 35

What of the following is a possible output for Z?

A) 2 **B)** 7 **C)** 12 **D)** 17 **E)** 22

```
int U, T, Z;
U = (int)(Math.random()*6 + 10);
T = (int)(Math.random()*5 + 3);
Z = (U * T)/10;
out.println(Z);
```

Question 36

Assume that a binary search tree using the letters B A C has a height of 1. Insert the letters to the right into an empty binary search tree. What is the height?

Note: Duplicate values **ARE** added to the binary tree. Duplicate values are added to the left of the original occurrence. This tree will have 16 nodes.

A) 4 **B)** 5 **C)** 6 **D)** 7 **E)** 8

```
U I L I N V I T A T I O N A L A
```

Question 37

What is the output of the code segment shown on the right?

A) false

B) true

```
boolean B = true;
for(int x = 12; x<=123; x+=3)
  B = B ^ true;
out.println(B);
```

Question 38

What is the output of the code segment shown on the right?

A) 60 **B)** 65 **C)** 70 **D)** 75 **E)** 80

```
int[][]Box = new int[10][10];
Box[0][0] = 2;
Box[0][1] = 3;
for (int x=1; x<10; x++)
  for (int y=0; y<10; y++)
    Box[x][y] = Box[x-1][y] + Box[0][1];
int T = 0;
for(int x = 0; x<=4; x++)
  T += Box[5][x];

out.println(T);
```

Question 39

The following operations are applied to an initially empty stack. At the end, four items remain on the stack. Find the sum of those items and write the number in answer blank #39.

```
push 10
push 8
push 12
pop
push 11
push 6
push 7
pop
pop
push 3
push 5
pop
```

Question 40

Evaluate the prefix expression to the right. Write the number in answer blank #40.

```
+ 8 * 2 / 60 + - 7 2 * 3 5
```

★ANSWER KEY – CONFIDENTIAL★

UIL COMPUTER SCIENCE – Invitational A 2024

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

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

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

Note: Correct responses are based on **Java SE Development Kit 20 (JDK 20)** from Oracle, 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 20 Standard Packages have been imported. Ignore any typographical errors and assume any undefined variables are defined as used.

Explanations:

1.	B	Start converting them all to base 2. You will notice quickly that A and B have different values. After converting C to base 2, you notice that choice B is the "different" choice. Converting the last two to binary proves that assumption.

2.	D	Use order of operations.
		Perform integer division first, then do modulus, then add the two values.
		`22 / 7 + 15 % 8`
		`3 + 7 = 10`

3.	C	The five operations will produce the following results:
		`22 + 8 = 30`
		`3 * 2 = 6`
		`7 / 3 = 2`
		`4 % 11 = 4`
		`12 - 3 = 9`
		There are no spaces between values on the same line.
		A new line will occur after the 6 and the 2.
		`306`
		`2`
		`49`

4.	C	`String St = "BACKPACK";` The length of St is 8. So T = 7 We are to print the char #6 in St which is the C

5.	A	`true && (true ||false)`
		Parentheses first - gives us
		`true && true = true`

6.	D	`Math.sqrt(40) = 6.32455` `Math.ceil()` rounds it up to the next highest "integer" and returns it as a double 7.0
7.	B	`double T = 21 / 2;;` T will have the value of 10.0 7.5 + 8.5 + 10.0 = 26.0

8.	E	`int M = 5;`
		`if(M > 10)` This if is false
		`{`
		`out.print(M);` Line not executed
		`M++;` Line not executed
		`}`
		`if(M < 3)` This if is false
		`out.print(M);` Line not executed
		`M++;` This line IS executed: M now is 6
		`if(M == 7)` This if is false
		`out.print(M);` Line not executed
		`out.print(M);` 6 is printed

9.	A	Since N is 5, the loop would do this:
		`for(int x = 5; x < 25; x=x+5)`
		`out.print(x+1 + " ");`
		While x has values 5 10 15 20
		It prints 6 11 16 21

10.	B	`int[] food = {5,2,3,4,1,0};`
		`out.print(food[food[0]-food[4]]);`
		`food[0]= 5`
		`food[4]= 1`
		`food[5-1] = food[4] = 1`

11.	A	M "grabs" the first value (12) from the Scanner String. Then, three values are "passed by." M will be the sum of the original 12 plus 56 plus 2 = 70

12.	D	`for(int i = 1; i <= 30; i = i + 2)`
		`  if (i%3==0)`
		`  out.print(i+" ");`
		The loop goes through all odd numbers 1 through 29.
		It only prints those odd numbers that are evenly divisible by 3.
		3 9 15 21 27

13.	B	`int w = 511;` w = 511 `w = w<<3;` w = 4088 `w = w>>2;` w = 1022 `w = w++ + w++;` w = 1022 + 1023 = 2045 `out.print(--w);` This prints 2044
14.	C	64 / 8 = 8

15.	B	`numbers.add(96);` `[96]`
		`for(int x=0; x<=100; x=x+12)`
		`  numbers.add(x);` `[96,0,12,24,36,48,60,72,84,96]`
		`for(int x=1; x<=5; x=x+1)`
		`  numbers.remove(0);` This removes the first 5 elements
		`out.println(numbers);` `[48,60,72,84,96]`

16.	A	`String yes = "012345678901234";` `int P = yes.indexOf("23",5);` P = 12 `String no = yes.substring(3,5);` no = "34" `out.println(no + P);` This prints 3412
17.	B	After 0 passes: x=1 y=20 After 1 pass : x=2 y=20 After 2 passes: x=4 y=19 After 3 passes: x=6 y=18 After 4 passes: x=8 y=17 After 5 passes: x=10 y=16 After 6 passes: x=12 y=15 After 7 passes: x=14 y=14 One more is added to x and one more is subtracted from y x=15 y=13 15 * 13 = 195

18.	B	`(12 | 9) ^ (14 & 10)`
		`(1100 OR 1001) XOR (1110 AND 1010)`
		`1101 XOR 1010`
		`0111 = 7`

19.	D	Fibonacci-esque `int[]Special = {2,5,7,12,19,31,50,81,131,212,343,...` The first two elements are 2 and 5. To get all other elements, add the two elements before that position.

20.	A	`for(int x=0; x<Moe.length; x++)`
		`  Moe[x] = x*x;`
		Moe is an array of the squares of the index.
		32*32 = 1024

21.	B	`for(int x=0; x<Larry.length; x++)`
		`  Larry[x] = Moe[x]%5;`
		Larry is an array of the perfect squares modulus 5
		11 * 11 % 5 = 1

22.	D	`for(int x=0; x<Shemp.length; x++)`
		`  Shemp[x] = Moe[x] - Larry[x];`
		Shemp is the square of the index minus the square of the index modulus 5
		16*16 - 16*16%5
		256 - 1 = 255

23.	A	`String ten = "JABAAAABBBBZAB";`
		`for(int x=1; x<=100; x++)`
		`  ten = ten.replaceAll("AB","");`
		`out.print(ten.length());`
		The first time replaceAll is used, three ABs are replaced.
		But after that happens, another AB is formed.
		The loop keeps removing those ABs and continues many more times than needed.
		The only thing left will be JZ - That is a length of 2

24.	C	Loop iterations x=0 T=6 x=2 T=7 x=4 T=8 x=6 T=9 x=8 T=10 Then x increments by 2, but does not do the loop body. The final value of T is 10

25.	D	`public static int Park(int A, int B)`
		`{`
		`  if (A==B)`
		`  return (A+B)/3;`
		`  if (A > B)`
		`  return Park(A-1, B) + A;`
		`  return Park(A, B-2) + B;`
		`}`
		Park(10,10) does not recurse … (10+10)/3 = 6

26.	D	Park(7,5) = Park(6,5) + 7 = 9+7= 16 Park(6,5) = Park(5,5) + 6 = 3+6 = 9 Park(5,5) = (5+5)/3 = 3
27.	A	Park(3,12) = Park(3,10) + 12 = 32+12= 44 Park(3,10) = Park(3,8) + 10 = 22+10= 32 Park(3,8) = Park(3,6) + 8 = 14+8= 22 Park(3,6) = Park(3,4) + 6 = 8+6= 14 Park(3,4) = Park(3,2) + 4 = 4+4= 8 Park(3,2) = Park(2,2) + 3 = 1+3= 4 Park(2,2) = (2+2)/3 = 1

28.	A	`low = []`
		`low.add(60)` `low=[60]`
		`low.add(75)` `low=[60,75]`
		`low.add(30)` `low=[30,75,60]`
		`low.add(12)` `low=[12,30,60,75]`
		`low.add(15)` `low=[12,15,60,75,30]`
		`low.add(90)` `low=[12,15,45,75,30,90,60]`
		`low.add(45)` `low=[12,15,45,64,30,90,60,75]`
		`low.remove()` The 12 is removed

29.	D	After the 12 is removed `low=[15,30,45,64,75,90,60]`

30.	E	Remove the 15
		`low=[30,60,45,64,75,90]`
		Remove the 30
		`low=[45,60,90,64,75]`

31.	D	frog = 0 x loop 1: frog loses 1, frog gains 1 x loop 2: frog loses 1, frog gains 2 x loop 3: frog loses 1, frog gains 3 … x loop 10: frog loses 1, frog gains 10 frog will equal 0+1+2+3+4+5+6+7+8+9 = 45
32.	A	Instantiating a Mom object sets the Name = "Bella" and the ID=33
33.	A	Instantiating a Son object with two parameters originally sets Name = "Bella" and the ID=33 Name is changed to Dylan, but the 55 is not used. ID is 33
34.	B	Instantiating a Son object with no parameters originally sets Name = "Bella" and the ID=33 The ID is set to 44 after the call to the parent (Mom).
35.	B	U has an integer value in the range [10,15] T has an integer value in the range [3,7] U*T has an integer value in the range [30,105] Z has an integer value in the range [3,10] 7 is the only choice in that range
36.	C	After building the tree: Level 0 contains U Level 1 contains I V Level 2 contains I L Level 3 contains I L N Level 4 contains AN T Level 5 contains A I T Level 6 contains A O Height is 6
37.	B	B starts out = true Every time it goes through the loop, it switches to the opposite boolean value. The loop iterates 38 times. Since this is an even number of iterations, the value will be the original one … true
38.	E	2 3 0 0 0 0 0 0 0 0 5 6 3 3 3 3 3 3 3 3 8 9 6 6 6 6 6 6 6 6 11 12 9 9 9 9 9 9 9 9 14 15 12 12 12 12 12 12 12 12 **17 18 15 15 15** 15 15 15 15 15 20 21 18 18 18 18 18 18 18 18 23 24 21 21 21 21 21 21 21 21 26 27 24 24 24 24 24 24 24 24 29 30 27 27 27 27 27 27 27 27 17+18+15+15+15 = 8

39.	32	Here is the evolution of the stack: Bottom -> Top
		`push 10 [10,8]`
		`push 8 [10,8]`
		`push 12 [10,8,12]`
		`pop [10,8]`
		`push 11 [10,8,11]`
		`push 6 [10,8,11,6]`
		`push 7 [10,8,11,6,7]`
		`pop [10,8,11,6]`
		`pop [10,8,11]`
		`push 3 [10,8,11,3]`
		`push 5 [10,8,11,3,5]`
		`pop [10,8,11,3]`
		10+8+11+3 = 32

40.	14	**+ 8 * 2 / 60 + - 7 2 * 3 5** **+ 8 * 2 / 60 + - 7 2 15** **+ 8 * 2 / 60 + 5 15** **+ 8 * 2 / 60 20** **+ 8 * 2 3** **+ 8 6** **14**