UIL COMPUTER SCIENCE WRITTEN TEST

2025 INVITATIONAL A

JANUARY 2025

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 – 2025 INVITATIONAL A

Note: Correct responses are based on **Java SE Development Kit 22 (JDK 22)** 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 22 Standard Packages have been imported. Ignore any typographical errors and assume any undefined variables are defined as used. **For all output statements, assume that the System class has been statically imported using:** **`import static java.lang.System.*;`**

Question 1

Which of the following is not equivalent to the expression $437_8 + 10101_2$?

A) 10310_4 **B)** 464_8 **C)** 134_{16} **D)** 100110100_2 **E)** All are equivalent

Question 2

What is output by the code to the right?

A) 55 **B)** 11 **C)** 7 **D)** 19

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

```
out.println(3+4 % 1+2 * 5+6);
```

Question 3

What is output by the code to the right?

A) 2.346 **B)** 2.34567

C) 2.34 **D)** 2.345

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

```
out.printf("%.3f",2.34567);
```

Question 4

What is output by the code to the right?

A) BackRiblis **B)** BackRilis

C) yBackRibilis **D)** yBackRiilis

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

```
String a = "Chilis";
String b = "BabyBackRibs";
b = b.substring(3, 11);
a = b + a.substring(2);
out.println(a);
```

Question 5

What is output by the code to the right?

A) true **B)** false

C) There is no output due to a syntax error.

```
boolean a = true;
boolean b = a ^ !a;
a = a | b & !a | !b;
out.println(a);
```

Question 6

What is output by the code to the right?

A) 3 **B)** 2.5 **C)** 3.0 **D)** 2.0

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

```
double a = 2.45;
out.println(Math.ceil(a));
```

Question 7

What is output by the code to the right?

A) 212 **B)** 112

C) 1212 **D)** 211

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

```
int i = 10;
if(i++ == 11)
    out.print(1);
else if(i++ == 11);
    out.print(2);
out.println(i);
```

Question 8

What is output by the code to the right?

A) 864 **B)** -14 **C)** 513 **D)** 14

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

```
int a = 17 - 8 * 3;
int b = a + 11 / 2;
out.println(a * b);
```

Question 9

How many `*`s are output by the code to the right?

A) `27` **B)** `33` **C)** `29` **D)** `30`

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

```
for(int y = 0; y < 12; y++)
   for(int c = 1; c < y; c *= 2)
       out.print("*");
out.println();
```

Question 10

What is the output by the code to the right?

A) `55`

B) `67`

C) `7`

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

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

```
int[] i = new int[] {
  17, 12, 9, 8, 39, 3
};
i[2] += i[4];
i[1] -= i[3];
int b = i[2];
b += i[1] + i[5];
out.println(b);
```

Question 11

Which of the following packages contains the `File` class?

A) `java.lang.*` **B)** `java.awt.*` **C)** `java.util.*` **D)** `java.io.*` **E)** None of the above.

Question 12

What is output by the code to the right?

A) `456` **B)** `106`

C) `561` **D)** `121`

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

```
int sum = 1;
for(int y = 0; y < 15; y++)
   for(int x = 0; x < y; x++)
       sum += x;
out.println(sum);
```

Question 13

What is the order of precedence for the operators to the right?

A) `II, IV, III, I` **B)** `IV, II, I, III`

C) `IV, II, III, I` **D)** `III, II, IV, I`

E) `II, IV, I, III`

```
I. || (logical)
II. ++ (post)
III. & (bitwise)
IV. -- (pre)
```

Question 14

What is output by the code to the right?

A) `8` **B)** `64` **C)** `16` **D)** `32`

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

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

Question 15

What is the output by the code to the right?

A) `[B, C, D]`

B) `[D, C, A]`

C) `[A, C, D]`

D) `[D, B, A]`

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

```
ArrayList<String> a;
a = new ArrayList<String>();
a.add("A");
a.add("B");
a.add("C");
a.remove(1);
a.add("D");
out.println(a);
```

Question 16

What is output by the code to the right?

A) `1234ABCD`

B) `[Ljava.lang.String;@156643d4`

C) Output cannot be determined until runtime.

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

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

```
String s = "1234ABCD";
char[]c = s.toCharArray();
out.println(c);
```

Question 17

What is output by the code to the right?

A) `X = 81` **B)** `X = X`

C) `0 = X` **D)** `X = 88`

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

```
char A = 'X';
int B = 81;
out.print(B < A ? A : 0);
out.print(" = ");
out.print(B > A ? B : A);
```

Question 18

What is output by the line marked `//q18` in the client code to the right?

A) `[5, 9, 13, 17, 25, 1]`

B) `[1, 5, 9, 13, 17, 25]`

C) `[17, 25, 1, 5, 9, 13]`

D) `[13, 17, 25, 1, 5, 9]`

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

Question 19

What is output by the line marked `//q19` in the client code to the right?

A) `[13, 21, 1, 9]`

B) `[17, 5, 212]`

C) `[13, 25, 1, 9]`

D) `[21, 25, 1, 9, 13]`

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

```
ArrayList<Integer> a;
a = new ArrayList<Integer>();
for(int y = 1; y < 30; y += 4)
    a.add(y);
Collections.rotate(a, -3);
a.remove(2);
a.remove(3);
out.println(a); //q18
a.add(212);
a.removeIf(x -> x % 3 == 2);
out.println(a); //q19
```

Question 20

What is output by the code to the right?

A) `117` **B)** `81`

C) `165` **D)** `80`

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

```
out.println(17 | 45 ^ 74 & 88);
```

Question 21

What is output by the line marked `//q21` in the client code to the right?

A) `8` **B)** `13`

C) `12` **D)** `5`

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

Question 22

What is output by the line marked `//q22` in the client code to the right?

A) `987` **B)** `128`

C) `465` **D)** `37`

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

Question 23

What is output by the line marked `//q23` in the client code to the right?

A) `7739` **B)** `616`

C) `2048` **D)** `28657`

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

```
public int recur(int i) {
    if(i < 0)
         return 1;
    if(i % 5 < 2)
         return recur(i - 2) +
                          recur(i - 3);
    else
         return recur(i - 2);
}
/////////////client code////////////
out.println(recur(10)); //q21
out.println(recur(32)); //q22
out.println(recur(51)); //q23
```

Question 24

What could replace **<1*>** in the code to the right so that the `A` class compiles and functions as intended?

A)
```
self.i = i;
self.s = s;
```
B)
```
this.i = i;
this.s = s;
```
C)
```
i = i;
s = s;
```
D) `super(i,s);`

E) More than one of the above.

Question 25

What could replace **<2*>** in the code to the right so that the `B` class compiles and functions as intended, intializing the `i` instance variable with value `7`?

A) `super(s, 7);`

B) `super.A(7, s);`

C) `super(7, s);`

D) `super.A(s, 7);`

E) `super();`

Question 26

What is the output by the line marked `//q26` in the client code to the right?

A) `4 10 10`

B) `4 10 8`

C) `4 8 8`

D) `3 7 7`

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

Question 27

What is the output by the line marked `//q27` in the client code to the right?

A) `c 10`

B) `c 16`

C) `c 14`

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

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

```
class A{

    int i;
    String s;

    public A(int i, String s) {
         <1*>
    }

    public int add() {
         return ++i;
    }

    public String toString() {
         return s+" "+i;
    }
}
class B extends A{

    public B(String s) {
         <2*>;
    }

    public int add() {
         i += 2;
         super.add();
         return i;
    }
}
///////////client code///////////
A a = new A(3, "a");
B b = new B("b");
A c = new B("c");
String o = "" + a.add();
o += " " + b.add();
o += " " + c.add();
out.println(o); //q26
c.add();
c.add();
out.println(c); //q27
```

Question 28

What is the output by the code to the right?

A) `true`

B) `false`

C) Output cannot be determined until runtime.

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

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

```
String s1 = "H3llo Th3r3!";
String s2 = "H..{2,4}\\S..{2,5}";
s1 = "" + s1.matches(s2);
out.println(s1);
```

Question 29

What could replace **<?*>** in the code to the right so that the code compiles and executes as intended?

A) `add` **B)** `push`

C) `append` **D)** A and B.

E) Any of the above.

Question 30

What is the output by the code to the right?

A) `[Purple, Orange, Red]`

B) `[Green, Yellow, Red]`

C) `[Purple, Red, Yellow]`

D) `[Blue, Purple, Yellow]`

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

```
Stack<String> stack;
stack = new Stack<String>();
stack.<?*>("Blue");
stack.<?*>("Purple");
stack.<?*>("Orange");
stack.pop();
stack.<?*>("Green");
stack.pop();
stack.<?*>("Yellow");
stack.<?*>("Red");
stack.pop();
out.println(stack);
```

Question 31

Assume that the elements to the right are inserted into an Unbalanced Binary Search Tree where duplicate elements are **not added** to the tree.

How many internal nodes will the tree have?

A) 15 **B)** 13

C) 10 **D)** 16

E) 9

Question 32

Under the same assumption as Question 31, how many leaf nodes will the tree have?

A) 1 **B)** 5

C) 3 **D)** 4

E) 7

Question 33

Under the same assumption as Question 31, what is the diameter of the tree?

A) 8 **B)** 6

C) 9 **D)** 10

E) 4

Question 34

Under the same assumption as Question 31, what is the worst-case time complexity for the operation `search()` in an Unbalanced Binary Search Tree? You may assume that n is the number of elements in the tree.

A) $O((n))$ **B)** $O(n^2)$

C) $O(n)$ **D)** $O(\sqrt{n})$

E) $O((n))$

```
34, 86, 28, 29, 33, 14, 52, 31,
92, 14, 15, 92, 31, 92, 105, 95,
97, 118
```

Question 35

Which of the following could replace **<1*>** to ensure that any classes that are stored as data within this data structure are compatible with the `Comparable` interface?

A) `extends` **B)** `implements`

C) `requires` **D)** Either A or B.

E) None of the above.

Question 36

Which of the following lines of code could replace **<2*>** so that the function `peek()` properly returns the value of the data stored in `head`?

A) `return head.data`

B) `return this.head.data`

C) `return this.head`

D) Either A or B.

E) All of the above.

Question 37

Which of the following well-known data structures is the class `DataStruct` an implementation of?

A) `LinkedList` **B)** `Queue`

C) `Stack` **D)** `Vector`

E) `Deque`

Question 38

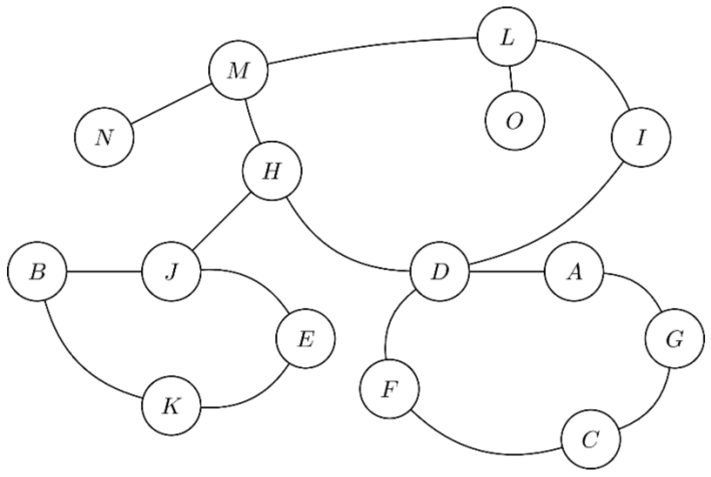
Which of the following classes would not be able to be stored within this data structure?

A) `Integer` **B)** `String`

C) `BigInteger` **D)** `double[]`

E) None of the above.

```
public class DataStruct<T <1*> Comparable<T>> {
     private class Node {
          public T data;

          public Node next;

          public Node(T d, Node n) {
               this.data = d;
               this.next = n;
          }
     }

     private Node head;

     private int size;

     public DataStruct() {
          this.size = 0;
          this.head = null;
     }

     public T peek() {
          if(head == null) {
               return null;
          }
          <2*>
     }

     public T pop() {
          T data = this.head.data;
          this.head = this.head.next;
          this.size--;
          return data;
     }

     public T push(T data) {
          Node newHead = new
               Node(data, this.head);
          this.head = newHead;
          this.size++;
          return data;
     }

     public int size() {
          return size;
     }
}
```

Question 39 Evaluate the following postfix expression. Assume that the ^ operator refers to the power operator, and that / is performed as integer division.	`252 42 36 - 2 ^ / -2 -34 89 + * +`
Question 40 Determine the longest simple cycle in the undirected graph to the right. Note that if multiple such solutions exist, chose the one that is lexicographically first.	

★ANSWER KEY – CONFIDENTIAL★

UIL Computer Science – 2024-2025 Invitational A

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

1) E	11) D	21) A	31) E
2) D	12) A	22) B	32) B
3) A	13) A	23) C	33) C
4) C	14) D	24) B	34) C
5) A	15) C	25) C	35) A
6) C	16) A	26) A	36) D
7) A	17) D	27) B	37) C
8) D	18) D	28) A	38) D
9) C	19) C	29) D	*39) -103
10) A	20) A	30) D	*40) A D F C G A

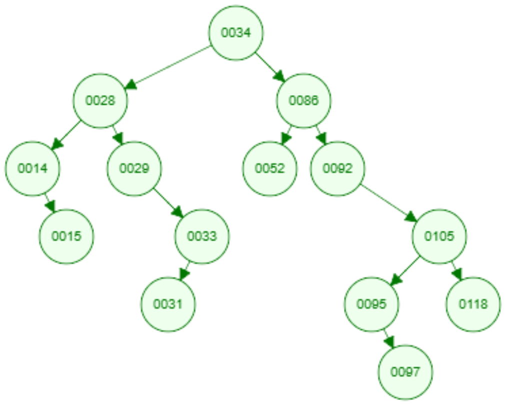
KEY

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

Note: Correct responses are based on **Java SE Development Kit 22 (JDK 22)** from Sun Microsystems, Inc. All provided code segments are intended to be syntactically correct, unless otherwise stated (e.g., "error" is an answer choice) and any necessary Java SE 22 Standard Packages have been imported. Ignore any typographical errors and assume any undefined variables are defined as used.

Explanations:

1.	E	All of the values are equivalent
2.	D	Simple order of operations problem
3.	A	Printf is formatting output, “`%.3f`” **rounds** the decimal value to 3 places
4.	C	Simple substring problem, substring is inclusive of first value, exclusive of last (if present)
5.	A	Simple Boolean solving
6.	C	`Math.ceil` returns next “whole” number as a double that is above the given value
7.	A	`i++` returns value of `i`, **THEN** increments, so value returned is one less than actual value
8.	D	Simple order of operations problem
9.	C	Simple math problem, tracing the loop
10.	A	Array problem, no tricks really just trace it out
11.	D	`File` class is in the `java.io` package
12.	A	Just trace the loop, you could also estimate and knock out the wrong answers
13.	A	Simple order of precedence
14.	D	Integer = 32 bits
15.	C	add to the end, Arraylists are 0-indexed. Trace it out.
16.	A	char arrays print out like a string, all other array types do not print legibly.
17.	D	In ternary, the char at the second term will be cast to an int if the first term is an int.
18.	A	Arraylist tracing, rotate will rotate the list like a circle.
19.	C	Arraylist tracing, removeIf removes a value if the expression is true for it.
20.	A	Simple bitwise tracing.
21.	A	Simple recursion tracing.
22.	B	Simple recursion tracing. There is a trick for this question, the answer will be equivalent to 2 ^ (n / 5 + 1).
23.	C	Simple recursion tracing. There is a trick for this question, the answer will be equivalent to 2 ^ (n / 5 + 1).
24.	B	`this.i` points to the instance variable, `i` points to the parameter in the constructor.
25.	C	`super(7, s)` is the only one that does not cause an error.
26.	A	For both of these question explanations I will refer to `i` as the value of each class instance. Both instances of the `B` class will be initialized with value `7`, and their `add` methods will add `3` every time they are called, giving both `b` and `c` a value of `10` (2 are added in the `add` method of class `B`, one is added in the `super` call to the `add` method of class `A`). The `A a` will have value `3`, and will add one when the `add` method is called, giving `a` a value of `4`. `c` is actually an instance of class `B`, because the `B` constructor is called when it is initialized.
27.	B	Since `c` is an instance of class `B`, and has a value of `10` after the code for question 26 (see the explanation for question 26). The `add` method is called twice so the value will be `16` after the code has executed. The `toString` method will return “`c 16`”, as “`c`” is the string `s` and the value is `16`.
28.	A	The value `true` is returned because the pattern described by `s2` is matched by the string `s1`. The pattern described by `s2` is as follows: the character `H`, followed by 3-5 characters (`.` means any character, do not need to be any specific character), followed by a non-whitespace character (`\\S`), followed by 3-6 characters (`.` means any character, do not need to be any specific character). `{a,b}` after a character means a match will be between `a` and `b` occurrences of that character.
29.	D	The methods `push` and `add` both will work to add a value to a stack.
30.	D	Stack tracing, First in First out.

31.	E	A copy of the binary search tree has been provided below:  0034 0028 0086 0014 0029 0052 0092 0015 0033 0105 0031 0095 0118 0097 Internal nodes are those nodes that have 1 or more children. There are a total of 9 nodes that have 1 or more child (nodes 34, 28, 14, 29, 33, 86, 92, 105, and 95).
32.	B	Leaf nodes are those nodes that have no children of their own. There are a total of 5 nodes that have no children of their own (nodes 15, 31, 52, 97, and 118).
33.	C	The diameter of a tree is the greatest number of edges between any two nodes within the tree. In the case of this tree, those nodes are 31 and 97, which have 9 edges between them.
34.	C	The worst-case scenario for an Unbalanced Binary Search Tree is that the nodes are inserted in sorted order. This effectively creates a Linked List, which has a linear time complexity for search operations.
35.	A	The key word `extends` is the only one among the ones listed which allows to specify which interfaces or classes the generic types must implement or inherit. This is commonly confused with the `implements` key word, which is used when declaring a class or interface that should express the behaviors of another interface. The `requires` key word, while a valid key word in the Java library, denotes a required library within a module, and thus is irrelevant to this question.
36.	D	Since there are no local versions of the variable `head`, either option A or option B will reference the global `head` variable and perform the function as intended. Option C will break since the return type of the function is of type `T` and not of provided type `Node`.
37.	C	This is a `Stack` since it denotes the methods `peek()`, `pop()`, and `push()` and has the FIFO (First-in-First-out) property with regards to how elements are handled. While the FIFO property *can* also be obtained using a `Deque`, for this to be a `Deque`, we would need to have separate methods for peeking, pushing, and popping elements from both sides of the `Queue`, which is not present in the provided implementation.
38.	D	`Integer`, `String`, and `BigInteger` all implement the `Comparable` interface. While `double[]` *can* actually be stored in a normal `Stack`, since `double[]` does not implement the `Comparable` interface, it cannot be stored in *this* implementation of a `Stack`.

39.	-103	The following is the process of evaluating the postfix expression (elements shown in parenthesis are the value pushed to the operand stack after performing the next operation):
		
		`252 42 36 – 2 ^ / -2 -34 89 + * +`
		`252 (6) 2 ^ / -2 -34 89 + * +`
		`252 (36) / -2 -34 89 + * +`
		`(7) -2 -34 89 + * +`
		`7 -2 (55) * +`
		`7 (-110) +`
		`(-103)`

40.	A D F C G A *or* ADFCGA	A simple cycle is a cycle in a graph with no repeated vertices (except the first to denote this as being a cycle and not a non-cyclic path) Note that to make a cycle as lexicographically small as possible, simply choose the node within the cycle with the lexicographically smallest label as the start of the cycle, and among its two (or more) neighbors, select the one that is lexicographically smallest. This ensures that the Three major simple cycles exist within the graph: 1. $B \rightarrow J \rightarrow E \rightarrow K \rightarrow B$ 2. $D \rightarrow H \rightarrow M \rightarrow L \rightarrow I \rightarrow D$ 3. $A \rightarrow D \rightarrow F \rightarrow C \rightarrow G \rightarrow A$ Of these, the largest simple cycle that is also the lexicographically smallest option is #3