UIL COMPUTER SCIENCE WRITTEN TEST

2025 INVITATIONAL B

FEBRUARY 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 B

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 12818_9 % 3657_{11}?

A) 1213_{10} **B)** 1273_{10} **C)** 6335_{10} **D)** 3903_{10} **E)** 816_{10}

Question 2

What is output by the code to the right?

A) 91 **B)** -3 **C)** 88 **D)** -91

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

```
out.println((-17 % 7) | (11 << 3));
```

Question 3

What is output by the code to the right?

A) \%f = %f **B)** \%f = 2.0

C) The code fails to compile when it attempts to resolve the escape sequence "\%".

D) The code fails to compile since unused arguments are passed to the `printf()` function.

E) The code fails to compile when it attempts to resolve the format conversion of "%%"

```
out.printf("\\%%f = \%f", 2, 'c');
```

Question 4

Assuming that indented lines are continuations of the previous line, what is output by the code to the right?

A) -6 **B)** -5 **C)** 5 **D)** 6

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

```
String s1 =
    "hEllo thEre... GEnEral KEnobi";
String s2 = s1.toLowerCase();
String[] arr1 = s1.split("E");
String[] arr2 = s2.split("E");
int diff = arr1.length -
    arr2.length;
out.println(diff);
```

Question 5

What is output by the code to the right?

A) true **B)** false

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

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

```
int num1 = 5, num2 = 6;
int num3 = (num1 & num2) >> 2;
boolean a = (boolean) num3;
boolean b = false;
out.println((a ^ !b) || (b & a));
```

Question 6

What is output by the code to the right?

A) 2 **B)** 2.0

C) 8 **D)** 8.0

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

```
long num1 = Math.floorDiv(7, 3);
double num2 = Math.pow(2, 3);
out.println(Math.min(num1, num2));
```

Question 7

Assuming that indented lines are continuations of the previous line, what is output by the code to the right?

A) `10 10 11`

B) `10 11 11`

C) `10 11 12`

D) `12 10 11`

E) `12 11 11`

```
int alpha = 10;
int beta = ++alpha;
int gamma = alpha++;
out.println(alpha + " " + beta
    + " " + gamma);
```

Question 8

What is output by the code to the right?

A) `none`

B) `Cum Laude`

C) `Magna Cum Laude`

D) `Summa Cum Laude`

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

```
double gpa = 3.961;
String latinHonors = "none";
switch(gpa) {
    case 3.731: {
        latinHonors = "Cum Laude";
        break;
    }
    case 3.886: {
        latinHonors = "Magna Cum Laude";
        break;
    }
    case 3.976: {
        latinHonors = "Summa Cum Laude";
        break;
    }
}
out.println(latinHonors);
```

Question 9

What is output by the code to the right?

A) `66 67 69 73 81`
`66 67 69 73 81`

B) `66 67 69 73 81`
`B C E I Q`

C) `B C E I Q`
`66 67 69 73 81`

D) `B C E I Q`
`B C E I Q`

E) None of the above.

```
for(int off = 1; off < 25; off *= 2) {
    out.print(('A' + off) + " ");
}
out.println();
for(int off = 1; off < 25; off *= 2) {
    out.print((off + 'A') + " ");
}
```

Question 10

Which of the following best describes the first issue with the code to the right? That is, which error, if any, is both accurate in its description and will cause the code to break at the earliest point possible?

A) `line 1` will cause a compilation error.

B) `line 2` will cause a compilation error.

C) `line 2` will cause a runtime error.

D) `line 3` will cause a runtime error.

E) `line 4` will cause a compilation error.

F) `line 4` will cause a runtime error.

G) None of the above.

```
int[] a = new int[] {}; // line 1
int[] b = new int[-1];  // line 2
a[0] = 1;               // line 3
b[-1] = 1;              // line 4
```

Question 11

Assume that the program for the code to the right is compiled and ran from the directory `/usr/uil/inv_b/written`, and assume that there is another file located at `/usr/uil/inv_b/written/in/q11.txt`. You may assume that the only other files that exist are those that are necessary to run and compile Java on an Operating System.

What is output by the code to the right?

A)
```
File 1 exists!
File 2 exists!
```

B)
```
File 1 exists!
File 2 does not exist...
```

C)
```
File 1 does not exist...
File 2 exists!
```

D)
```
File 1 does not exist...
File 2 does not exist...
```

E) `line 1` will cause a runtime error (`IOException`).

F) `line 2` will cause a runtime error (`IOException`).

```
File f1 = new File("in/q11.txt"); // line 1
if(f1.exists()) {
   out.println("File 1 exists!");
} else {
   out.println("File 1 does not exist...");
}
File f2 = new File("q11.txt"); // line 2
if(f2.exists()) {
   out.println("File 2 exists!");
} else {
   out.println("File 2 does not exist...");
}
```

Question 12

You may assume that the line commented with `// cont.` is a continuation of the line above it.

What is output by the code to the right?

A) `4` **B)** `26` **C)** `32` **D)** `36` **E)** `59`

```
int[] arr = new int[] {1, 2, 3, 4,
    5, 6, 7, 8, 9, 10}; // cont.
int[] pre = new int[arr.length];
pre[0] = arr[0];
for(int i = 1; i < arr.length; i++) {
    pre[i] = pre[i-1] + arr[i];
}
out.println(pre[7] - pre[3] + arr[9]);
```

Question 13

What is output by the code to the right?

A) `-1` **B)** `0` **C)** `1` **D)** `4`

E) `1073741823`

```
out.println(~3 & 4 >> 2);
```

Question 14

What is output by the code to the right?

A)
```
Option 1
```

B)
```
Option 2
```

C)
```
Option 1
Option 3
```

D)
```
Option 2
Option 4
```

E) The program terminates successfully without creating any output.

```
int a = Integer.MAX_VALUE;
int b = Integer.MIN_VALUE;
int c = a + 1;

if(c >= a)
    out.println("Option 1");
else if(c >= b);
    out.println("Option 2");

if(c == a)
    out.println("Option 3");
else if(c == b)
    out.println("Option 4");
```

Question 15

Which of the following can replace **<?*>** so that the code to the right compiles and produces the output `"[2.3, 1]"`?

A) `Integer` **B)** `Double`

C) `Comparable`

D) `? extends Comparable`

E) None of the above.

```
ArrayList< <?*> > ratings;
ratings = new ArrayList< <?*> >();

ratings.add(2.3); ratings.add(1);
out.println(ratings);
```

Question 16

What is output by the line marked `// line 3` in the client code to the right? You may assume that all mono-spaced lines among the answer choices that are indented are continuations of the previous line.

A) `[Point@372f7a8d, Point@2f92e0f4,`
`Point@28a418fc, Point@5305068a]`
(Four memory addresses determined at runtime)

B) `[(-2.300000,-1.000000),`
`(1.000000,2.000000),`
`(2.300000,2.000000),`
`(9.000100,2.300000)]`

C) `[(-2.3,-1.0), (1.0,2.0), (2.3,2.0),`
`(9.0001,2.3)]`

D) `[(1.0,2.0), (2.3,2.0), (9.0001,2.3),`
`(-2.3,-1.0)]`

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

Question 17

Which of the following changes to the code to the right would cause the line marked `// line 3` to output the following:

`[(-2.3,-1.0), (1.0,2.0), (2.3,2.0),`
`(9.0001,2.3)]`

A) Change the code for `// line 3` to instead be the code fragment in the section labeled "`Option A`".

B) Add the code fragment in the section labeled "`Option B`" to be a method inside of the class Point.

C) Add the code fragment in the section labeled "`Option C`" to be a method inside of the class Point.

D) More than one of the choices above.

E) No change is required since the line marked `// line 3` already outputs the requested text.

F) None of the above since none resolve the compile error.

```
class Point {
    public double x, y;

    public Point(double x, double y) {
        this.x = x;
        this.y = y;
    }
}

class SortByX implements Comparator<Point> {
    public int compare(Point p1, Point p2) {
        int comp = Double.compare(p1.x, p2.x);
        return comp == 0 ?
            Double.compare(p1.y, p2.y) : comp;
    }
}

// Q18.A

//////////////////client code////////////////
ArrayList<Point> points = new
ArrayList<Point>();
points.add(new Point(1, 2));
points.add(new Point(2.3, 2));
points.add(new Point(9.0001, 2.3));
points.add(new Point(-2.3, -1));

SortByX sortAlg = new SortByX();   // line 1
Collections.sort(points, sortAlg); // line 2
out.println(points);               // line 3

//////////////////Option A//////////////////
List<String> list;
list = points.stream().map(p ->
        String.format("(%f,%f)", p.x, p.y)
    ).toList();
out.println(list);                 // line 3

//////////////////Option B//////////////////
public String toString() {
    return "(" + x + "," + y + ")";
}

//////////////////Option C//////////////////
public String toString() {
    return String.format("(%f,%f)", x, y);
}
```

Question 18

Building off of the code from question 16 and question 17, which of the following will allow for the `ArrayList` named `points` to instead be sorted by Y-coordinate, with ties broken by X-coordinate and printed to the console?

A) Replace the line labeled `// Q18.A` with the code fragment in the section to the right labeled "`Option A`" and replace all instances of "`SortByX`" to "`SortByY`" on the line labeled `// line 1`.

B) Replace the line labeled `// line 2` with the code fragment in the section to the right labeled "`Option B`"

C) Replace the line labeled `// line 2` with the code fragment in the section to the right labeled "`Option C`"

D) A and C.

E) All of the above.

```
//////////////////Option A//////////////////
class SortByY implements Comparator<Point> {
    public int compare(Point p1, Point p2) {
        int comp = Double.compare(p1.y, p2.y);
        return comp == 0 ?
            Double.compare(p1.x, p2.x) : comp;
    }
}

//////////////////Option B//////////////////
Collections.sort(points, (p1, p2) -> {
    int comp = Double.compare(p1.y, p2.y);
    return comp == 0 ?
        Double.compare(p1.x, p2.x) : comp;
});

//////////////////Option C//////////////////
Collections.sort(points, (Point p1, Point p2) -> {
    int comp = Double.compare(p1.y, p2.y);
    return comp == 0 ?
        Double.compare(p1.x, p2.x) : comp;
});
```

Question 19

The code fragment labeled "`Option A`" in question 17, and the code fragment labeled "`Option C`" in question 18 are all examples of what concept in the java programming language?

A) Method/Namespace Referencing.
B) Functional Interfaces.
C) Anonymous Inner Classes.
D) Lambda Expressions.
E) None of the above.

Question 20

What is output by the code to the right?

A) `[Alice, Bob, TrUdy]`
B) `[ALICE, BOB, TRUDY]`
C) `[A, B, TU]`
D) There is no output due to a compile error.
E) There is no output due to a runtime error.

```
List<String> names = List.of(
    "Alice", "Bob", "TrUdy");
List<String> uppercase = names.stream()
    .map(String::toUpperCase)
    .collect(Collectors.toList());
out.println(uppercase);
```

Question 21

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

A) `3`
B) `-1`
C) `1`
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) `9`
B) `19`
C) `-3`
D) `10`
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) `0`
B) `-59`
C) `-1`
D) `-5`
E) There is no output due to a runtime error.

```
int recur(int a, int b) {
    if(a == b)
        return 1;
    if(a + b <= 0)
        return -1;
    if(a < b)
        return 2 + recur(a, b - 3);
    return -2 + recur(a / 2, b);
}
//////////////client code//////////////
out.println(recur(12, 14)); //q21
out.println(recur(32, 45)); //q22
out.println(recur(74, 14)); //q23
```

Question 24

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

A) `public`

B) `protected`

C) `private`

D) Nothing is required

E) More than one of the above

Question 25

Assume that **<?*>** has been filled in properly, what is the output by the lines marked `//q25` in the client code to the right?

A) `8080`

B) `80808080`

C) `808080`

D) `808017`

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

Question 26

Assume that **<?*>** has been filled in properly, what is the output by the line marked `//q26` in the client code to the right?

A) `RoarRoarRoarRoar`

B) `RoarRoarRoarRoarRoar`

C) `RoarRoarRoar`

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

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

Question 27

Assume that **<?*>** has been filled in properly, what is the output by the line marked `//q27` in the client code to the right?

A) `55`

B) `75`

C) `79`

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

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

```
interface Animal {
    String roar();
}

class Cat implements Animal {
    int age, speed;

    public Cat(int a, int s) {
        age = a;
        speed = s;
    }

    <?*> String roar() {
        return "Roar";
    }

    public void run() {
        out.print(speed);
    }

    public int birthday() {
        return age++;
    }
}

class Cheetah extends Cat {
    public Cheetah(int a) {
        super(a, 80);
    }

    public void run() {
        super.run();
        super.run();
    }
}
//////////////client code///////////////
Animal a = new Cat(14, 15);
Cat b = new Cat(13, 17);
Cheetah c = new Cheetah(23);
Cat d = new Cheetah(19);
c.run(); //q25
d.run(); //q25
String f = a.roar();
f += b.roar() + c.roar();
f += d.roar();
out.println(f); //q26
int i = a.birthday();
i += b.birthday();
i += c.birthday();
i += d.birthday();
out.println(i); //q27
```

Question 28

What is the output by the code to the right?

A) `true true`

B) `false true`

C) `true false`

D) `false false`

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

```
String s = "Luke Skywalker";
String fin = "";
String r = "(\\w)+|(\\s){0,3}";
fin += s.matches(r) + " ";
r = "([A-z]+ ?)*";
fin += s.matches(r);
out.println(fin);
```

Question 29

What could replace **<1*>** in the code to the right so that the code compiles and executes as intended?

A) `Queue<String>`

B) `List<String>`

C) `LinkedList<>`

D) A and B.

E) None of the above.

Question 30

What could replace **<2*>** in the code to the right so that the code compiles and executes as intended?

A) `poll`

B) `pop`

C) `remove`

D) A and C.

E) Any of the above.

Question 31

What is the output by the code to the right?

A) `Green White Gray`

B) `Red Green Purple`

C) `Red Purple Yellow`

D) `Green Blue Black`

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

```
Queue<String> q;
q = new <1*>();
q.offer("Red");
q.offer("Green");
String s = q.<2*>();
q.offer("Purple");
q.offer("Blue");
q.offer("Orange");
q.offer("White");
s += " " + q.<2*>();
q.offer("Yellow");
q.offer("Brown");
q.offer("Black");
q.offer("Gray");
s += " " + q.<2*>();
System.out.println(s);
```

Question 32

What could replace **<1*>** in the code to the right so that the code compiles and `e` is put into the next available space in `arr`, and `size` represents the current size of the structure?

A) `arr[size] = e`

B) `arr[size++] = e`

C) `arr[++size] = e`

D) A and B.

E) More than one of the above.

Question 33

What could replace **<2*>** in the code to the right so that the code compiles and the contents of `arr` are copied to `s`?

A) `System.arraycopy(arr, 0, s, 0, size)`

B) `System.arraycopy(arr, s, 0, 0, size)`

C) `System.arraycopy(arr, 0, size, s, 0, size)`

D) `System.arraycopy(arr, s, size)`

E) `System.arraycopy(arr, s, 0, size)`

Question 34

What could replace **<3*>** in the code to the right so that the code compiles and the value of the instance variable len is printed by the line marked `//q35`?

A) `s.len`

B) `s.getLen()`

C) `Structure.len`

D) A and C.

E) Any of the above.

Question 35

What is the output by the line marked `//q35` in the client code to the right?

A) `5`

B) `8`

C) `16`

D) `10`

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

Question 36

What is the output by the line marked `//q36` in the client code to the right?

A) `[212, Exactly, Knows, SpiderMan]`

B) `[212, Exactly, Nobody, SpiderMan]`

C) `[Correct, Exactly, Knows, SpiderMan]`

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

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

```
class Structure<E> {
   E[] arr;
   int size, len;

   public Structure() {
      arr = (E[])(new Object[1]);
      size = 0;
      len = 1;
   }

   public void add(E e) {
      if(size == len)
         resize();
      <1*>;
   }

   public E remove(int i) {
      E[] s = (E[])(new Object[size - 1]);
      for(int j = 0; j < i; j++)
         s[j] = arr[j];
      for(int j = i + 1; j < size; j++)
         s[j - 1] = arr[j];
      E e = arr[i];
      arr = s;
      size--;
      len = arr.length;
      return e;
   }

   public void resize() {
      E[] s = (E[])(new Object[size * 2]);
      <2*>;
      arr = s;
      len = arr.length;
   }

   public String toString() {
      return Arrays.toString(arr);
   }
}
///////////////client code///////////////
Structure<String> s;
s = new Structure<String>();
s.add("212");
s.add("Purple");
s.add("Correct");
s.remove(1);
s.add("Exactly");
s.remove(0);
s.add("Nobody");
s.add("Knows");
s.add("SpiderMan");
out.println(<3*>); //q35
s.remove(2);
out.println(s); //q36
```

Question 37

Which of the following is not a legal Java statement?

A) `Object o0o0 = new TreeMap<ArrayList, HashSet>();`

B) `Collection c__c = new HashSet<Queue>();`

C) `BigInteger bbno$ = (BigInteger)(BigInteger.ZERO);`

D) `List<Object> l1st = new ArrayList<>();`

E) All statements are legal.

Question 38

What is the output by the code to the right?

A) 298 **B)** 392

C) 326 **D)** 360

E) There is no output due to an infinite loop.

```
int sum = 0;
for(int i = 0; i < 16; i++)
    for(int j = i; j >= i / 2; j--)
        sum += j / 3 * 2;
out.println(sum);
```

Question 39

What is the 8-bit two's complement representation of the following decimal number?

-103_{10}

Question 40

What is the in-order traversal of the binary search tree created by inserting the following values in order?

34 67 212 17 9 6 104 8 29 48 97 147 1

★ANSWER KEY – CONFIDENTIAL★

UIL COMPUTER SCIENCE – 2024-2025 INVITATIONAL B

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

1) D	11) B	21) A	31) B
2) B	12) D	22) B	32) B
3) C	13) B	23) C	33) A
4) C	14) D	24) A	34) A
5) C	15) C	25) B	35) B
6) B	16) A	26) A	36) C
7) E	17) B	27) D	37) E
8) E	18) E	28) B	38) C
9) A	19) D	29) C	*39) 10011001
10) C	20) B	30) D	*40) See Explanation

** 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.	D	12818_9 % $3657_{11} \equiv 8684_{10}$ % $4781_{10} = 3903_{10}$
2.	B	-17 % 7 = -3 = $11111111111111111111111111111101_2$ 11 << 3 = 88 = $00000000000000000000000001011000_2$ -3 | 88 = -3
3.	C	The string "\\" is the escape sequence to print the character '\', and "%%" is the format for printing the '%' character in formatted strings/outputs. Therefore, "\\%%f" resolves to the string "\%f". Even though the string "%f" appears in the formatted string, note that this does not get filled with a floating-point number – this only happens in the format specifier string, which gets resolved from left-to-right. Some websites might incorrectly make reference to "\%" being an alternative way to print a '%' within a `printf` statement; however, this has never been an actual feature of Java and will cause a compile-time error since "\%" is not one of the approved escape sequences.
4.	C	`arr1` has a length of `6`, while `arr2` only has a length of `1`. Their difference is `5`.
5.	C	Note that even though `num3` resolves to the value of `1`, which can be represented with a single bit of a boolean, this conversion is not supported in Java using the built-in type casting operator.

6.	B	`num1 = 2`
		`num2 = 8.0`
		Since we have a `double` as one of the arguments, this will force the compiler to use the version of the `min` method which takes in two `double`s and produces a `double`. This causes the output to be `2.0` even though the input of `2` ended up being the number that was the minimum between `2` and `8.0`.

7.	E	Note that this tests the difference between the pre- and post-increment operator. The pre-increment operator performs the increment and then returns the new value. The post increment operator returns the existing value and then performs the increment. In either operator, the original value is incremented.
8.	E	Note that only convertible `int` values, `String`s or `enum` variables are permitted to be the argument of a `switch` statement. Thus, a compile-time error occurs.
9.	A	Note that in either case of the character appearing first, or the integer appearing first, since `int` requires more space than `char`, Java will automatically cast the result to the larger datatype (`int`), which is why only integer values of the characters are printed. This can be circumvented by casting the result back to a `char`, but simply re-arranging the operators does nothing.
10.	C	All statements will compile just fine. The first statement to cause an error will be line 2 since it attempts to allocate negative space, which is not allowed. Lines 3 and 4 will both cause runtime errors, but since the question asks for the first error, the correct answer is C.
11.	B	Java can use both absolute and relative pathing; however, since the paths included in both line 1 and line 2 do not begin with a '/' or a drive letter, the default behavior is to use relative pathing. Since we are told that the base path of the program is `/usr/uil/inv_b/written` and we know that the file `/usr/uil/inv_b/written/in/q11.txt` exists, the first file's path will resolve to `/usr/uil/inv_b/written + in/q11.txt = /usr/uil/inv_b/written/in/q11.txt`, which exists. However, the same cannot be said for the second since the path it will resolve to is `/usr/uil/inv_b/written + q11.txt = /usr/uil/inv_b/written/q11.txt` which we are not explicitly told exists.
12.	D	`pre = {1, 3, 6, 10, 15, 21, 28, 36, 45, 55}` `pre[7] - pre[3] + arr[9] ≡ 36 - 10 + 10 = 36`

13.	B	Order of precedence is ~, then >>, then &.
		`~3 = -4`
		`4 >> 2 = 1`
		`-4 & 1 = 0`

14.	D	Note that when you add one to `2147483647` (`Integer.MAX_VALUE`) when stored as an `int`, instead of becoming `2147483648`, it wraps around to `-2147483648` (`Integer.MIN_VALUE`).

15.	C	Selecting `Integer` causes the `add(2.3)` to error and selecting `Double` causes the `add(1)` to error. Both `Integer` and `Double` extend the `Comparable` interface. While wildcards (`?`) can be used within the first set of generics, they cannot be used within the second when instantiating an object.
16.	A	Since no `toString` method is present within the `Point` class, memory addresses are printed.
17.	B	While all three options will compile and run just fine, both `Option A` and `Option C` will cause the x- and y-coordinate values to be printed to 5 decimal places of precision, while only `Option B` will allow for the flexibility of having variable precision.
18.	E	`Option A` is equivalent to the original solution for sorting in question 16. `Option B` and `Option C` are identical except for the fact that `Option C` specified the input type. Lambda expressions are valid regardless of whether the data type of the inputs are specified. All options will provide the same outcome and are equivalent.
19.	D	While options A through D are all valid concepts in the Java programming language, the examples presented are all examples of Lambda Expressions. Method referencing allows you to reference the function implementation of an already existing method. Anonymous inner classes was the traditional way of implementing Lambda-Expression-like functionality before their introduction. Lambda Expressions rely on the existence of Functional Interfaces, they are not equivalent to one another.
20.	B	This is an example that uses the Method Referencing operator "`::`" which allows you to reference the function implementation of an already existing method. The code present applies the `String.toUpperCase()` method on all `String`s contained within the `names ArrayList` and then outputs the new list to the `uppercase List`. Printing them out gives the result of running `toUpperCase()` on each `String` in the original list.
21.	A	Simple recursive tracing
22.	B	Simple recursive tracing
23.	C	Simple recursive tracing
24.	A	The method is defined without a scope identifier in the interface, meaning it is set to default access, or "package-protected". When implementing a method previously defined in either an abstract class or interface, you cannot decrease the scope, meaning it needs to be public, as neither protected or private scope is wider scope than "package-protected".
25.	B	The run method of the `Cat` class will print `80`, but the overridden method of the `Cheetah` class will print `8080`. Both `c` and `d` are instantiated as instances of the `Cheetah` class meaning both will print `8080`, hence the answer is B.
26.	A	Each call to method `roar` will return `Roar`, so the answer is `RoarRoarRoarRoar`
27.	D	The `birthday` method is not defined for interface `Animal`, so calling `birthday` on `a` will result in a compile error (the compiler only looks at what each instance is defined as, not what it is instantiated as).
28.	B	`(\\w)+|(\\s){0,3}` is a regular expression that will match a string of all word characters (letters, digits, and underscores) or 0-3 instances of whitespace characters, which will not match the string "`Luke Skywalker`". `([A-z]+ ?)*` is a regular expression that will match 0 or more occurences of more than one character with ASCII value between `A` and `z`, followed by 0 or 1 space, which does match the string "`Luke Skywalker`".
29.	C	LinkedList is the only option that is not an interface, and therefore the only option that can be instantiated.
30.	D	`LinkedList` can use `poll` or `remove` method to get an item out of the front of the queue (`LinkedList` is a `Queue`).
31.	B	Simple trace the queue, first in first out add to the end remove from the beginning.
32.	B	Since arrays are 0 indexed, the last index in the array that is being used is `size - 1`, so you would use `size++` to put the element at the current last index in the list, and increment `size` once to then to set it equal to the number of items in the array.
33.	A	`System.arraycopy` takes the following parameters in the following order: `array1`, `start_index1`, `array2`, `start_index2`, `length`. So A is the only option that makes sense.

34.	A	Instance variable `len` is not static, so C will not work. There is no `getLen()` method so B will not work, leaving only A.
35.	B	The maximum amount of elements in the structure was 5, and the size of the array is doubled each time there are not enough spaces, meaning that the array size will always be the smallest power of 2 that is greater than the maximum amount of elements from the array, which would be 8.
36.	C	Simply trace like it is an ArrayList, removing from the given index locations.
37.	E	All 4 of these are legal instantiations (type them into a compiler and they will all work).
38.	C	Simple mathematics tracing.
39.	10011001	Do 103 in binary: `1100111` Add a 0 in the end: `01100111` Flip all the bits: `10011000` Add 1: `10011001`
40.	`1 6 8 9 17 29 34 48 67 97 104 147 212`	
	In order for binary search trees is just the sorted order, so you don't even need to make the tree.	