# UIL COMPUTER SCIENCE WRITTEN TEST

# 2018 STATE

## MAY 2018

## 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)
```

# UIL COMPUTER SCIENCE WRITTEN TEST – 2018 STATE

**Note:** Correct responses are based on **Java SE Development Kit 8 (JDK 8)** 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 8 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.**

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

**A)** `10` **B)** `11` **C)** `5` **D)** `6` **E)** `-11`

```
out.print(0b10101010%0b00001111);
```

**Question 2.**

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

**A)** `21` **B)** `19` **C)** `10` **D)** `3` **E)** `15`

```
out.println(5%4+8-2*3);
```

**Question 3.**

What is the output of the code segment to the right? Asterisks indicate blank spaces.

**A)** `**85.46`

**B)** `85.46973`

**C)** `85.4697`

**D)** Error. Will not compile.

**E)** Error. Throws an IllegalFormatConversionException.

```
out.printf("%7.5s","85.4697294");
```

**Question 4.**

What is the output of the code segment shown here?

```
String s="feels like summer";
String t=s.substring(s.indexOf("u")).concat(s.substring(1, 4));
out.print(t);
```

**A)** `mmereel`

**B)** `ummereel`

**C)** `mmerfeel`

**D)** `eelummer`

**E)** `ummerfeel`

**Question 5.**

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

**A)** `true` **B)** `false`

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

**Question 6.**

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

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

```
out.print(Math.round(Math.E));
```

**Question 7.**

Given the code segment shown on the right, which of the following additional lines of code will compile and execute correctly?

**A)** double a=w+x+y+z;

**B)** float b=w+x+y+z;

**C)** long c=w+x+y+z;

**D)** int d=w+x+y+z;

**E)** More than one of the above.

```
int w=9;
long x=8;
double y=3.5;
float z=4.15f;
```

**Question 8.**

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

**A)** A

**B)** AC

**C)** C

**D)** B

**E)** D

```
int m=10,n=-8,p=2;
if(m<n&&p==n/2)
  if(m>0||p==2)
    out.print("A");
  else
    out.print("B");
else
  if(n<=-8^p*m==20)
    out.print("C");
  else
    out.print("D");
```

**Question 9.**

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

**A)** thginllap

**B)** upallnight

**C)** thginllapu

**D)** hginllapu

**E)** There is no output due to an error.

```
String s="upallnight";
int i=s.length();
do {
     out.print(s.charAt(i));
     i--;
}while(i>0);
```

**Question 10.**

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

**A)** 0

**B)** 10

**C)** 5

**D)** 11

**E)** 6

```
boolean[] ba=new boolean[10];
for(int x=1;x<ba.length;x+=2)
  ba[x]=true;
int y=0;
for(int x=0;x<ba.length;x++)
  if(ba[x])
    y++;
out.print(y);
```

**Question 11.**

Which of the following must replace **<code>** to ensure that the main method will compile and that when executed the user can type in their first and last name from the keyboard?

**A)** Scanner s=new Scanner(System);

**B)** Scanner s=new Scanner(in);

**C)** Scanner s=new Scanner(new File("in");

**D)** Scanner s=new Scanner("System.in");

**E)** None of the above.

```
import static java.lang.System.*;
import java.io.File;
import java.io.IOException;
import java.util.Scanner;

public class Q11 {

public static void main(String[] args) {
     <code>
     out.print("First Name: ");
     String fn=s.next();
     out.print("Last Name: ");
     String ln=s.next();
     }
}
```

**Question 12.**

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

**A)** -47

**B)** 47

**C)** 37

**D)** -37

**E)** -10

```
int s=0;
for(int x=20;x>0;x--) {
     if(x%3==0)
          s-=x;
     if(x%2!=0)
          s+=x;
     }
out.print(s);
```

**Question 13.**

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

**A)** -5

**B)** -6

**C)** 5

**D)** 6

**E)** Error. Will not compile.

```
int x=-5;
x=~-++x;
out.print(x);
```

**Question 14.**

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

**A)** 8

**B)** 65536

**C)** 32

**D)** 16

**E)** 64

```
out.print(Character.SIZE);
```

**Question 15.**

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

**A)** Dumas

**B)** -1

**C)** true

**D)** false

**E)** There is no output due to an error.

```
ArrayList<String> a=new
ArrayList<String>();
a.add("Dalhart");
a.add("Dumas");
a.add("Muleshoe");
a.add("Earth");
a.remove(1);
out.println(a.remove("Dumas"));
```

**Question 16.**

Which of the following cannot be modified using `final`?

**A)** class

**B)** constructor

**C)** method

**D)** field

**E)** All of the above can be modified with `final`.

**Question 17.**

When the code segment shown on the right has been executed which of the following statements is true about the set `s`?

**A)** `s` contains only even numbers from 20 to 38 inclusive.

**B)** `s` contains only odd numbers from 21 to 39 inclusive.

**C)** `s` contains 1000 different random numbers.

**D)** `s` contains all of the numbers from 20 to 38 inclusive.

**E)** `s` contains only even numbers from 10 to 40 exclusive.

```
Random r=new Random();
Set<Integer> s=new TreeSet<Integer>();
for(int x=1;x<=1000;x++)
     s.add((r.nextInt(10)+10)*2);
```

**Question 18.**

Which of the following methods will return the greatest common divisor of `a` and `b`?

**I.**

```
public static int gcd(int
a,int b) {
  while(b!=0) {
    int t=b;
    b=a%b;
    a=t;
  }
  return a;
}
```

**II.**

```
public static int gcd(int
a,int b) {
  while(a!=b)
    if(a>b)
      a=a-b;
    else
      b=b-a;
  return a;
}
```

**III.**

```
public static int gcd(int
a,int b) {
  int d=Math.max(a, b);
  for(;d>=1;d--)
    if(a%d==0&&b%d==0)
      break;
  return d;
}
```

**A)** `I`

**B)** `II`

**C)** `III`

**D)** `I` and `II`

**E)** `I, II` and `III`

**Question 19.**

Which of the following can correctly replace **<code>** in the code segment shown on the right?

**A)** `nextInt()`

**B)** `nextDouble()`

**C)** `next()`

**D)** `nextFloat()`

**E)** More than one of the above.

```
Scanner s=new Scanner("2 12 6 8 1");
int sum=0;
while(s.hasNext())
     sum+=s.<code>;
out.print(sum);
```

**Question 20.**

What is the output of the code segment shown here?

```
String s="uil.academics@uiltexas.org";
out.print(s.matches(".+"));
out.print(s.matches("\\S{3}.\\D+@[a-z]+.\\w{3}"));
out.print(s.matches("uil\\.?\\S*@+uiltexas.org"));
```

**A)** `falsefalsetrue`

**B)** `falsetruetrue`

**C)** `truefalsetrue`

**D)** `truetruetrue`

**E)** `falsefalsefalse`

**Question 21.**

Which of the following classes is immutable?

**A.**

```
import java.util.*;
public class ImmutableClass {
  private int a,b;
  private Stack<String> st;

  public ImmutableClass(int a,int  b,
  Stack<String> st) {
    this.a=a;
    this.b=b;
    this.st=st;
  }

  public int getA() {return a;}

  public int getB() {return b;}

  public Stack<String> getST(){
    Stack<String> t=new Stack<String>();
    t.addAll(st);
    return t;
  }
}
```

**B.**

```
import java.util.*;
public class ImmutableClass {
  public int a,b;
  public Stack<String> st;

  public ImmutableClass(int a,int  b,
  Stack<String> st) {
    this.a=a;
    this.b=b;
    this.st=st;
  }

  public void setA(int a){this.a=a;}

  public void setB(int b){this.b=b;}

  public int getA() {return a;}

  public int getB() {return b;}

  public Stack<String> getST(){
    Stack<String> t=new Stack<String>();
    t.addAll(st);
    return t;
  }
}
```

**C.**

```
import java.util.*;
public final class ImmutableClass {
  private final int a,b;
  private final Stack<String> st;

  public ImmutableClass(int a,int  b,
  Stack<String> st) {
    this.a=a;
    this.b=b;
    this.st=st;
  }

  public int getA() {return a;}

  public int getB() {return b;}

  public Stack<String> getST(){
    Stack<String> t=new Stack<String>();
    t.addAll(st);
    return t;
  }
}
```

**D.**

```
import java.util.*;
public final class ImmutableClass {
  private final int a,b;
  private final Stack<String> st;

  public ImmutableClass(int a,int  b,
  Stack<String> st) {
    this.a=a;
    this.b=b;
    this.st=st;
  }

  public int getA() {return a;}

  public int getB() {return b;}

  public Stack<String> getST(){return st;}
}
```

E. More than one of the classes shown above is immutable.

**Question 22.**

Which line in the code shown here contains an error?

```
public class MyClass {

        public static void main(String[] args) {
                System.out.println(myMethod(1,6,"My dog has fleas."));//line #1
        }

        public static String myMethod(int i,int j,String s) {
                String t="";
                int x=i;
                int y=j;//line #2
                for(int i=x;i<y;i++) {//line #3
                        t+=s.substring(i, i+1);
                }
                return t;//line #4
        }
}
```

**A)** line #1 **B)** line #2 **C)** line #3 **D)** line #4 **E)** None of the above. There are no errors.

**Question 23.**

What is the output of **line #1** in the code segment shown on the right?

**A)** `{102=4, 225=1, 299=2, 312=3, 541=2}`

**B)** `{225=1, 541=2, 299=2, 312=3, 102=4}`

**C)** `{225=1, 541=2, 102=4, 312=3, 299=2}`

**D)** `{102, 225, 299, 312, 541}`

**E)** `{1, 2, 2, 3, 4}`

**Question 24.**

What is the output of **line #2** in the code segment shown on the right?

**A)** 2

**B)** 3

**C)** 312

**D)** 299

**E)** 300

```
//Code for questions 23 and 24.

TreeMap<Integer,Integer> tm=new
TreeMap<Integer,Integer>();
tm.put(225, 1);
tm.put(541, 2);
tm.put(102, 4);
tm.put(312, 3);
tm.put(299, 2);
out.println(tm);//line #1
int i=tm.ceilingEntry(300).getValue();
out.print(i);//line #2
```

**Question 25.**

What is printed by the main method shown on the right?

**A)** 45

**B)** 20

**C)** 43

**D)** 13

**E)** 25

```
public static void main(String[] args) {
  long n=4;
  out.print(f(n));
  }
public static long f(long n) {
  if(n<=1)
    return 1;
  else
    return 5+f(n-1)+f(n-2);
  }
```

**Question 26.**

Which of the following best describes the error within the interface shown on the right?

**A)** The signature must contain the keyword `implements`.

**B)** The interface does not contain a constructor.

**C)** The area and perimeter methods must be implemented.

**D)** `Shape` must be declared as `final` and contain fields that are declared as `final`.

**E)** There are no errors in the code shown. The interface `Shape` will compile and execute as intended.

```
public interface Shape {
     public double area();
     public double perimeter();
}
```

**Question 27.**

Which of the following statements is false?

**A)** An object cannot be instantiated from an abstract class.

**B)** It is possible to define an abstract class that does not contain any abstract methods.

**C)** A subclass can extend multiple abstract classes.

**D)** A subclass can be abstract even if its superclass is concrete.

**E)** A class that contains abstract methods must be abstract.

**Question 28.**

What is the run time efficiency (Big O value) of the segment of code shown on the right?

**A)** O(n log n)

**B)** O(x log y)

**C)** O(log n)

**D)** $O(n^2)$

**E)** $O(n^x \log y)$

```
for(int x=0;x<n;x+=4)
     for(int y=1;y<n;y*=3)
          out.print("Big O");
```

**Question 29.**

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

**A)** `47`

**B)** `27`

**C)** `33`

**D)** `35`

**E)** Error. Throws an ArrayIndexOutOfBoundsException.

```
int[][][] a=
{{{3,2,1},{1,2,3},{6,5,4}},
{{4,5,6},{9,5,1},{7,5,3}},
{{8,5,2},{1,2,4},{8,4,3}}};
int sum=0;
for(int x=0;x<3;x+=2)
     for(int y=0;y<3;y+=2)
          for(int z=0;z<3;z+=2)
               sum+=a[x][y][z];
out.print(sum);
```

**Question 30.**

When `n` is a power of 2, which of the following is always equivalent to `m%n`?

**A)** `m&n`

**B)** `m&(n-1)`

**C)** `m|(n-1)`

**D)** `m^n`

**E)** `m^(n-1)`

```
//Use the following code to answer questions 31, 32 and 33.
public static void sort(int[] list, int startIndex)
  {
  if ( startIndex >= list.length - 1 )
    return;
  int minIndex = startIndex;
  for ( <code 1> )
    if (list[index] < list[minIndex] )
      minIndex = index;
  int temp = list[startIndex];
  list[startIndex] = list[minIndex];
  list[minIndex] = temp;
  sort(<code 2>);
  }
```

**Question 31.**

Which of the following would best replace **<code 1>** in the sort method implemented above to ensure that list is sorted in ascending order?

**A)** `int index = 0; index < list.length-1; index++`

**B)** `int index = startIndex - 1; index < list.length; index++`

**C)** `int index = startIndex + 1; index < list.length; index++`

**D)** `int index = startIndex + 1; index < minIndex; index++`

**E)** `int index = startIndex + 1; index < list.length-1; index++`

**Question 32.**

Which of the following should replace **<code 2>** in the sort method implemented above to ensure that list is sorted in ascending order?

**A)** `startIndex+1, list`

**B)** `list, startIndex`

**C)** `list`

**D)** `list, startIndex + 1`

**E)** `startIndex`

**Question 33.**

Assuming that **<code 1>** and **<code 2>** have been replaced with correct code, which sorting algorithm does the sort method shown above implement?

**A)** bubble sort

**B)** merge sort

**C)** quick sort

**D)** insertion sort

**E)** selection sort

```
//Use the UILString class to answer questions 34 and 35.

public class UILString <code> Comparable<UILString>{

      private String str;

      public UILString(String s) {str=s;}

      public String toString() {return str;}

      public int compareTo(UILString o) {
            int t=0;
            int i1=str.length()-1;
            int i2=o.toString().length()-1;
            while((t==0)&&(i1>=0)&&(i2>=0)) {
                  char c1=str.charAt(i1);
                  char c2=o.toString().charAt(i2);
                  t=c1-c2;
                  i1--;i2--;
            }
            return t;
      }
}
```

**Question 34.**

Which of the following should replace **<code>** in the class shown above?

**A)** implements

**B)** inherits

**C)** extends

**D)** iterator

**E)** comparator

**Question 35.**

Assuming that **<code>** has been filled in correctly, what is the output of the following client code?

```
List<UILString> list=new LinkedList<UILString>();
list.add(new UILString("monkey"));
list.add(new UILString("Monkey"));
list.add(new UILString("Zebra"));
list.add(new UILString("monkEy"));
list.add(new UILString("monkeY"));
list.add(new UILString("zebra"));
Collections.sort(list);
out.print(list);
```

**A)** [Monkey, monkEy, monkeY, monkey, Zebra, zebra]

**B)** [Zebra, zebra, monkeY, monkEy, Monkey, monkey]

**C)** [monkey, Monkey, monkEy, zebra, Zebra, monkeY]

**D)** [Zebra, zebra, Monkey, monkEy, monkeY, monkey]

**E)** [monkeY, Zebra, zebra, monkEy, Monkey, monkey]

**Question 36.**

If the binary tree shown on the right is traversed in a postorder fashion, the result is a ____________ expression.

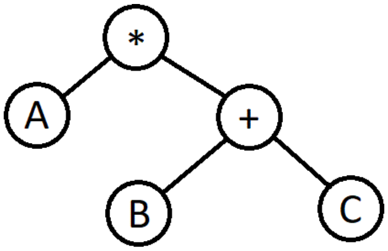
**A)** binary

**B)** prefix

**C)** infix

**D)** postfix

**E)** A postorder traversal does not result in a valid expression.



**Question 37.**

The class `Edge` and the accompanying client code shown on the right are intended to model a graph data structure. What must replace **<code>** within the client code segment to ensure it will compile and execute correctly?

**A)** `new ArrayList();`

**B)** `graph.add(new Edge());`

**C)** `graph.add(new ArrayList<Edge>());`

**D)** `graph=new ArrayList<Edge>();`

**E)** No additional code is required.

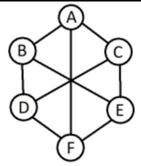
```
//Use the following to answer questions 37 and 38.

public class Edge {
  char o,d;
  public Edge(char o, char d) {
    this.o=o;
    this.d=d;}
}
//client code
List<ArrayList<Edge>> graph=new
ArrayList<>();
<code>
graph.get(0).add(new Edge('A','B'));
graph.get(0).add(new Edge('A','C'));
<code>
graph.get(1).add(new Edge('B','A'));
graph.get(1).add(new Edge('B','D'));
graph.get(1).add(new Edge('B','E'));
<code>
graph.get(2).add(new Edge('C','A'));
graph.get(2).add(new Edge('C','D'));
<code>
graph.get(3).add(new Edge('D','B'));
graph.get(3).add(new Edge('D','C'));
<code>
graph.get(4).add(new Edge('E','B'));
graph.get(4).add(new Edge('E','F'));
<code>
graph.get(2).add(new Edge('F','E'));
```

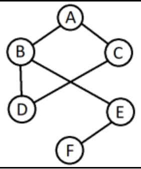
**Question 38.**

Once completed correctly, which of the following graphs does the client code shown on the right model?

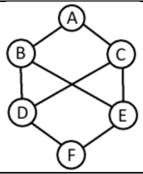
**A)**



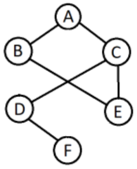
**B)**



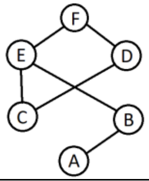
**C)**



**D)**



**E)**

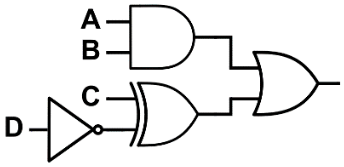


**Question 39.**

If both values are shown using signed 8-bit 2's complement representation, what is 11101101 minus 11100100? Write your answer using signed 8-bit 2's complement notation.

**Question 40.**

Write the Boolean expression represented by the diagram shown on the right. You must write your answer using generic notation. Express all operators explicitly and do not use Java operators.



# ★ANSWER KEY – CONFIDENTIAL★

## UIL COMPUTER SCIENCE – 2018 STATE

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

1) C

2) D

3) A

4) B

5) A

6) D

7) A

8) E

9) E

10) C

11) B

12) C

13) A

14) D

15) D

16) B

17) A

18) E

19) E

20) D

21) C

22) C

23) A

24) B

25) E

26) E

27) C

28) A

29) D

30) B

31) C

32) D

33) E

34) A

35) E

36) D

37) C

38) B

*39) 00001001

*40) $A * B + C \oplus \overline{D}$

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

**Note:** Correct responses are based on **Java SE Development Kit 8 (JDK 8)** 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 8 Standard Packages have been imported. Ignore any typographical errors and assume any undefined variables are defined as used.

Explanations:

<table>
<tr><td>1.</td><td>C</td><td>The leading 0b indicates that both values are binary. $10101010_2 = 170_{10}$. $00001111_2 = 15_{10}$. 170 % 15 = 5.</td></tr>
<tr><td>2.</td><td>D</td><td>5 % 4 + 8 – 2 * 3 =<br>1 + 8 – 6 =<br>9 – 6 =<br>3</td></tr>
<tr><td>3.</td><td>A</td><td>The format specifier %7.5s places the first five characters from the corresponding string value right justified in seven spaces.</td></tr>
<tr><td>4.</td><td>B</td><td><table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td></tr>
<tr><td>f</td><td>e</td><td>e</td><td>l</td><td>s</td><td></td><td>l</td><td>i</td><td>k</td><td>e</td><td></td><td>s</td><td>u</td><td>m</td><td>m</td><td>e</td><td>r</td></tr>
</table>
The index of "u" is 12. s.substring(12) is "ummer". s.substring(1,4) is "eel".<br>"ummer"+"eel" = "ummereel"</td></tr>
<tr><td>5.</td><td>A</td><td>(T^T)==((T||T)&&!(T&&T))<br>F==T&&!T<br>F==T&&F<br>F==F<br>T</td></tr>
<tr><td>6.</td><td>D</td><td>Math.E returns the mathematical constant e which is an irrational number approximately equal to 2.71828. Math.round rounds its argument to the nearest whole number.</td></tr>
<tr><td>7.</td><td>A</td><td>When y is included in the expression, w, x and z are all promoted to double and the expression evaluates to a double. This value can be assigned to the double variable a. An explicit cast would be required to assign the sum to the other three variables.</td></tr>
<tr><td>8.</td><td>E</td><td>10 < - 8 && 2 == -8 / 2<br>False && False<br>False<br>Skip to the else statement.<br>-8 <= -8 ^ 2 * 10 == 20<br>True ^ True<br>False<br>Skip to the else statement.<br>Print "D".</td></tr>
<tr><td>9.</td><td>E</td><td>i gets 10. Throws a StringIndexOutOfBoundsException.</td></tr>
<tr><td>10.</td><td>C</td><td><table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td></tr>
<tr><td>F</td><td>T</td><td>F</td><td>T</td><td>F</td><td>T</td><td>F</td><td>T</td><td>F</td><td>T</td></tr>
</table>
The default value for Boolean is false. The code sets every other element to true beginning with index 1 and then counts the number of true values in the array.</td></tr>
<tr><td>11.</td><td>B</td><td>The default input stream is the keyboard. The System class has been imported so it does not need to be included in the statement.</td></tr>
<tr><td>12.</td><td>C</td><td>x=20 x%3=2 x%2=0 s=0<br>x=19 x%3=1 x%2=1 s=19<br>x=18 x%3=0 x%2=0 s=1<br>x=17 x%3=2 x%2=1 s=18<br>x=16 x%3=1 x%2=0 s=18<br>x=15 x%3=0 x%2=1 s=18<br>x=14 x%3=2 x%2=0 s=18<br>x=13 x%3=1 x%2=1 s=31<br>x=12 x%3=0 x%2=0 s=19<br>x=11 x%3=2 x%2=1 s=30<br>x=10 x%3=1 x%2=0 s=30<br>x=9 x%3=0 x%2=1 s=30<br>x=8 x%3=2 x%2=0 s=30<br>x=7 x%3=1 x%2=1 s=37<br>x=6 x%3=0 x%2=0 s=31<br>x=5 x%3=2 x%2=1 s=36<br>x=4 x%3=1 x%2=0 s=36<br>x=3 x%3=0 x%2=1 s=36<br>x=2 x%3=2 x%2=0 s=36<br>x=1 x%3=1 x%2=1 s=37</td></tr>
</table>

|  |  |  |
|---|---|---|
| 13. | A | ~ - ++ (-5) =<br>~ - (-4) =<br>~ 4 =<br>- 5 |
| 14. | D | Character.SIZE returns the number of bits used to represent a char value in unsigned binary form, which is 16. |
| 15. | D | a.remove(1) removes the string "Dumas" from the arraylist. An attempt to remove a non-existent object using a.remove("Dumas") returns false. |
| 16. | B | Classes, methods and fields can all be declared as final. A constructor cannot be declared to be final. |
| 17. | A | r.nextInt(10) returns a random whole number between 0 (inclusive) and 10 (exclusive). Adding 10 yields a value between 10 and 19 (inclusive). Multiplying by 2 gives just the even numbers between 20 and 38 (inclusive). A Set does not allow duplicates so there will be at most 10 values in the set since there are on 10 even numbers between 20 and 38 (inclusive). |
| 18. | E | I. Euclid's algorithm using division.<br>II. Euclid's algorithm using subtraction.<br>III. Brute force method checking every possible divisor. |
| 19. | E | next() returns a string and therefore will not compile. Answer choices A, B and D are each auto un-boxed and will compile. |
| 20. | D | ".+" matches any character one or more times → true<br>"\\S{3}.\\D+@[a-z]+.\\w{3}" matches exactly 3 non whitespace characters, any character, one or more non-digits, a @, one or more lower case letters, any character, and exactly 3 word characters. → true<br>"uil\\.?\\S*@+uiltexas.org" matches uil, a period once or not at all, a non-whitespace character zero or more times, a @ one or more times, followed by uiltexas.org. → true |
| 21. | C | Objects instantiated from an immutable class cannot be changed. The rules for creating an immutable class are:<br>• There should be no setter methods.<br>• All fields should be final and private.<br>• The class should be declared as final.<br>• Do not return a reference to a mutable object<br>In answer choice A the class and the fields are not final. Answer choice B has public fields, the class is not final and it contains setter methods. Answer choice D returns a reference to a mutable object. |
| 22. | C | Declaring i as the loop control variable creates a duplicate identifier because i has already been used as a name for one of the parameters of the method. |
| 23. | A | A map uses keys and entries. In tm.put(225, 1), 225 is the key and 1 is the entry. A TreeMap stores its data sorted on the keys. |
| 24. | B | ceilingEntry returns "a key-value mapping associated with the least key greater than or equal to the given key, or null if there is no such key". The smallest key value greater than 300 is 312. The entry value associated with 312 is 3. |
| 25. | E | Here is a printout of the call stack:<br>n=4 5<br>n=3 5<br>n=2 5<br>n=1 1<br>n=0 1<br>n=1 1<br>n=2 5<br>n=1 1<br>n=0 1<br>25 |
| 26. | E | At a minimum, an interface must only present the signatures of those methods that must be implemented in any classes that implement the interface. |
| 27. | C | No class, abstract or not, can extend more than one other class. |

<table>
<tr><td>28.</td><td>A</td><td>Outer loop executes ¼ n times. Inner loop executes $\log_3(n)$ times. Ignore the constant ¼ and the base 3 to get O(n log n).</td></tr>
<tr><td>29.</td><td>D</td><td>3 + 1 + 6 + 4 + 8 + 2 + 8 + 3 = 35</td></tr>
<tr><td>30.</td><td>B</td><td>Let n = 4.<br>Let m = 15.<br>15 % 4 = 3<br>15 & (4 – 1) = 15 & 3<br>$1111_2 = 15_{10}$<br>$0011_2 = 3_{10}$<br>1111 & 0011 = 0011<br>$0011_2 = 3_{10}$</td></tr>
<tr><td>31.</td><td>C</td><td>This is a selection sort. The loop begins at the first element of the unsorted portion of the list and searches for the smallest element left in the remainder of the list. When the loop terminates the smallest remaining element is  swapped with the first element in the unsorted portion of the list and the starting place is incremented by one. This particular implementation makes a recursive call until there is only one element left in the unsorted portion of the list.</td></tr>
<tr><td>32.</td><td>D</td><td>The recursive call to sort passes the partially sorted list and increments the starting point.</td></tr>
<tr><td>33.</td><td>E</td><td>See #31.</td></tr>
<tr><td>34.</td><td>A</td><td>Comparable is an interface. Interfaces must be implemented.</td></tr>
<tr><td>35.</td><td>E</td><td>Sorts alphabetically from right most character to the left. Remember, B comes before a.</td></tr>
<tr><td>36.</td><td>D</td><td>A postorder traversal of this tree yields A B C + * which is a postfix expression.</td></tr>
<tr><td>37.</td><td>C</td><td>This is an adjacency edge list. For each vertex there is a list of edges. A new ArrayList must be added for each vertex where all of the edges associated with that vertex may be stored.</td></tr>
<tr><td>38.</td><td>B</td><td>Answer choices A, C, D and E all show an edge from D to F. There is never a D to F Edge object instantiated in the client code.</td></tr>
<tr><td>39.</td><td>00001001</td><td>
<table>
<tr><td></td><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td></tr>
<tr><td>~</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>+</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>=</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td>=</td><td>-19</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td></tr>
<tr><td>~</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td>+</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>=</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td></tr>
<tr><td>=</td><td>-28</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>
-19 – (-28) = -19 + 28 = 9 = 00001001<br><br><b><i>Answer must contain 8 bits.</i></b></td></tr>
<tr><td>40.</td><td>$\underline{A * B + C \oplus \overline{D}}$</td><td>Also accept $C \oplus \overline{D} + A * B$. Do not accept any Java operators.</td></tr>
</table>