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`

```java
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`

```java
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.

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

Question 4.

What is the output of the code segment shown here?

```java
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`

```java
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`

```java
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.

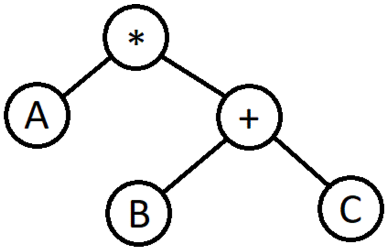
A) binary

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.

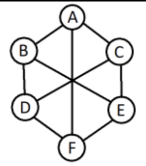
```
//Use the following to answer questions 37 and 38.

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'));
```

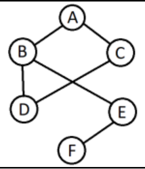
Question 38.

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

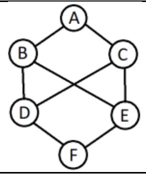
A)



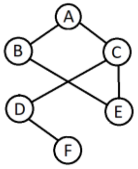
B)



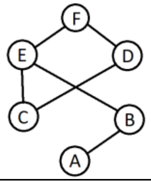
C)



D)



E)

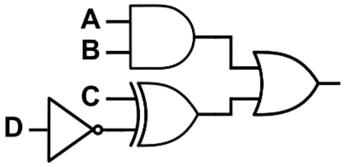


Question 39.

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 =
1 + 8 – 6 =
9 – 6 =
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".
"ummer"+"eel" = "ummereel"</td></tr>
<tr><td>5.</td><td>A</td><td>(T^T)==((T||T)&&!(T&&T))
F==T&&!T
F==T&&F
F==F
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
False && False
False
Skip to the else statement.
-8 <= -8 ^ 2 * 10 == 20
True ^ True
False
Skip to the else statement.
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
x=19 x%3=1 x%2=1 s=19
x=18 x%3=0 x%2=0 s=1
x=17 x%3=2 x%2=1 s=18
x=16 x%3=1 x%2=0 s=18
x=15 x%3=0 x%2=1 s=18
x=14 x%3=2 x%2=0 s=18
x=13 x%3=1 x%2=1 s=31
x=12 x%3=0 x%2=0 s=19
x=11 x%3=2 x%2=1 s=30
x=10 x%3=1 x%2=0 s=30
x=9 x%3=0 x%2=1 s=30
x=8 x%3=2 x%2=0 s=30
x=7 x%3=1 x%2=1 s=37
x=6 x%3=0 x%2=0 s=31
x=5 x%3=2 x%2=1 s=36
x=4 x%3=1 x%2=0 s=36
x=3 x%3=0 x%2=1 s=36
x=2 x%3=2 x%2=0 s=36
x=1 x%3=1 x%2=1 s=37</td></tr>
</table>

13.	A	~ - ++ (-5) = ~ - (-4) = ~ 4 = - 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. II. Euclid's algorithm using subtraction. 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 "\\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 "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: • There should be no setter methods. • All fields should be final and private. • The class should be declared as final. • Do not return a reference to a mutable object 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:
		n=4 5
		n=3 5
		n=2 5
		n=1 1
		n=0 1
		n=1 1
		n=2 5
		n=1 1
		n=0 1
		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.
Let m = 15.
15 % 4 = 3
15 & (4 – 1) = 15 & 3
$1111_2 = 15_{10}$
$0011_2 = 3_{10}$
1111 & 0011 = 0011
$0011_2 = 3_{10}$</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

<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>

Conference____________________________ Code Number____________________________

UIL COMPUTER SCIENCE WRITTEN TEST

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

1) ________

2) ________

3) ________

4) ________

5) ________

6) ________

7) ________

8) ________

9) ________

10) ________

11) ________

12) ________

13) ________

14) ________

15) ________

16) ________

17) ________

18) ________

19) ________

20) ________

21) ________

22) ________

23) ________

24) ________

25) ________

26) ________

27) ________

28) ________

29) ________

30) ________

31) ________

32) ________

33) ________

34) ________

35) ________

36) ________

37) ________

38) ________

39) ____________

40) ____________

FOR ADMINISTRATIVE USE ONLY

# Right:	×	6 pts	=	
# Wrong:	×	-2 pts	=	
# Skipped:	×	0 pts	=	0

	Score	Initials
Judge #1:		
Judge #2:		
Judge #3:		

Computer Science Competition
State 2018
Programming Problem Set

I. General Notes

1. Do the problems in any order you like. They do not have to be done in order from 1 to 12.

2. All problems have a value of 60 points.

3. There is no extraneous input. All input is exactly as specified in the problem. Unless specified by the problem, integer inputs will not have leading zeros. Unless otherwise specified, your program should read to the end of file.

4. Your program should not print extraneous output. Follow the form exactly as given in the problem.

5. A penalty of 5 points will be assessed each time that an incorrect solution is submitted. This penalty will only be assessed if a solution is ultimately judged as correct.

II. Names of Problems

Number	Name
Problem 1	Ankur
Problem 2	Catalina
Problem 3	Dennis
Problem 4	Dinesh
Problem 5	Emma
Problem 6	Johnny
Problem 7	Konstantinos
Problem 8	Leonardo
Problem 9	Milo
Problem 10	Oscar
Problem 11	Richa
Problem 12	Romero

1. Ankur

Program Name: Ankur.java Input File: none

Ankur has just learned about transcendental numbers and wants to output these variations of the two most well-known mathematical constants in various formats, as shown below. Can you help him write a program to do that? *Here's a hint that might be helpful:* ***e, f,*** *and* ***g****.*

Input: None

Output: These six values formatted exactly as shown.

Expected output:

```
3141.59
3141.592654
3.141593e+03
271.828
271.828183
2.718282e+02
```

2. Catalina

Program Name: Catalina.java Input File: catalina.dat

Catalina loves acronyms and has discovered that several everyday words commonly used are acronyms. For example, in the military, **AWOL** means **Absent WithOut Leave**, and **SNAFU** is used in a slightly sarcastic and cynical way to say things aren't really going great, as usual, standing for **Situation Normal All Fouled Up**. She decides to write a program that will take the phrase and reduce it to its acronym, taking only the significant letters, which she uppercases in the actual input sentence, and then parses through the sentence and forms the acronym from those letters.

Input: Several phrases, each on one line, with key letters uppercased.

Output: The resulting one-word acronym made up of the uppercased letters in the phrase, eliminating spaces and all other letters and symbols.

Sample input:

```
Absent WithOut Leave
Self-Contained Underwater Breathing Apparatus
Completely Automated Public Turing test to tell Computers and Humans Apart
Situation Normal, All Fouled Up
Junior Reserve Officer Training Corps
Missing In Action
```

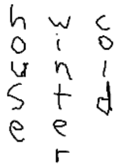
Sample output:

```
AWOL
SCUBA
CAPTCHA
SNAFU
JROTC
MIA
```

3. Dennis

Program Name: Dennis.java Input File: dennis.dat

Last winter on a very cold day Dennis was looking at the icicles hanging from the eve of his house and he began to wonder what words would look like if they were icicles. He got a piece of scratch paper and tried it out. He rewrote the words "**house**", "**winter**" and "**cold**" as if they were icicles hanging from his house. It came out like this:



As you can see, Dennis' penmanship is quite poor. So, to make his word icicles look better, Dennis decides he needs a program to read his words and print the icicles.

Input: A file that contains several sets of words to be printed as if they were icicles. The first line will contain a value S that represents the number of sets of words followed by S sets of words. For each set there will be a line that contains a number N representing the number of words in the set followed by N words. There are no blank lines between sets of words.

Output: Each set of words printed side by side in columns spelled from the top down. There should be as many rows as there are characters in the longest word and no more. Any space in a column not taken up by a letter in a word must be a space character. Each set should be followed by a row of asterisks (*) exactly as wide as the number of columns used to display the set. There should not be any blank lines between the sets.

Sample input:

```
2
3
house
winter
cold
5
cat
turtle
aardvark
dog
fish
```

Sample output:

```
hwc
oio
unl
std
ee
 r
***
ctadf
auaoi
trrgs
 td h
 lv
 ea
  r
  k
*****
```

4. Dinesh

Program Name: Dinesh.java Input File: dinesh.dat

In the recent advanced algorithms unit, Dinesh has learned about the concept of shortest path, where there is an undirected graph of nodes with each edge a certain distance between a pair of nodes, and where the task is to find the shortest distance required to travel between the two nodes.

For example, here is a picture of a graph, and the data that represents it.

```
ONE----4---TWO
| \          |
|   \        |
20    14     8
|       \    |
|         \  |
THREE--5---FOUR
```

The first line of data represents the four nodes of the graph, followed by a value N, then followed by N edge connections between two nodes, stated as the two names followed by an integer representing the distance between that pair of nodes. Following the edge designations is another value M, followed by M pairs of nodes. Between each pair we want to know the shortest path.

The first one is easy since they are adjacent. The distance between ONE and TWO is 4. The next one is a bit more complex since there are three different ways to travel between ONE and THREE. The direct route has a distance of 20, but the route through node FOUR is 19, and even better the route that goes through TWO and FOUR has a distance of 17, clearly the shortest path, even though it seems to be the long way around.

Please help Dinesh as he struggles to code the solution to this problem.

------sample data set-----
```
ONE TWO THREE FOUR
5
ONE TWO 4
ONE THREE 20
THREE FOUR 5
ONE FOUR 14
TWO FOUR 8
5
ONE TWO
ONE THREE
ONE FOUR
THREE FOUR
THREE TWO
```

Input: First will be listed an integer G, indicating G graph data sets to follow. Each graph data set will consist of a row of node names, all on one line, all uppercased, with single space separation. On the next line will be an integer N, followed on the next N lines by a pair of nodes and an integer indicating the distance between that pair of nodes. Finally, an integer M will be followed by M pairs of nodes, between which the shortest path is to be determined.

Output: The pair of nodes in question followed by the shortest path between that pair of nodes, in the exact format shown in the sample output below.

(Continued on next page)

4. Dinesh (continued)

Sample input:

```
2
ONE TWO THREE FOUR
5
ONE TWO 4
ONE THREE 20
THREE FOUR 5
ONE FOUR 14
TWO FOUR 8
5
ONE TWO
ONE THREE
ONE FOUR
THREE FOUR
THREE TWO
ALPHA BETA GAMMA DELTA EPSILON
6
ALPHA BETA 1
ALPHA GAMMA 7
ALPHA EPSILON 3
BETA EPSILON 6
GAMMA EPSILON 2
EPSILON DELTA 3
3
ALPHA DELTA
BETA GAMMA
EPSILON BETA
```

Sample output:

```
ONE to TWO:4
ONE to THREE:17
ONE to FOUR:12
THREE to FOUR:5
THREE to TWO:13
ALPHA to DELTA:6
BETA to GAMMA:6
EPSILON to BETA:4
```

5. Emma

Program Name: Emma.java **Input File: emma.dat**

Emma likes box-like patterns that can be produced with different computer algorithms and has designed one she thinks is pretty interesting. She needs your help though because her skills aren't quite ready to tackle the pattern she created. She decided to base it on a single integer value, like 3, and then create a box three times that size, a 9x9 box, which contained an X pattern of stars in the middle and four 3x3 boxes of stars in each corner. For lack of a better term, she decides to call it the **X-Box star pattern**. It looks like this:

```
***   ***
***   ***
***   ***
   * *
    *
   * *
***   ***
***   ***
***   ***
```

For an input value 4, the pattern would be similar, but slightly different, like this:

```
****    ****
****    ****
****    ****
****    ****
    *  *
     **
     **
    *  *
****    ****
****    ****
****    ****
****    ****
```

Input: Several integers N, such that 0<N<=10, all on one line with single space separation.

Output: An X-Box pattern as shown above, characterized by a grid three times the size of N, in other words, 3Nx3N, with an NxN box of stars in each corner, connected by an X pattern in the middle. Each complete output will be followed by the line "**==========**".

Sample input:

```
3 4 5
```

(continued on next page)

5. Emma (continued)

Sample output:

```
***   ***
***   ***
***   ***
   * *
    *
   * *
***   ***
***   ***
***   ***
==========
****    ****
****    ****
****    ****
****    ****
    *  *
     **
     **
    *  *
****    ****
****    ****
****    ****
****    ****
==========
*****     *****
*****     *****
*****     *****
*****     *****
*****     *****
     *   *
      * *
       *
      * *
     *   *
*****     *****
*****     *****
*****     *****
*****     *****
*****     *****
==========
```

6. Johnny

Program Name: Johnny.java Input File: johnny.dat

A huge Star Trek fan for as long as he can remember, Johnny is fascinated with the Universal Translator used to help with communications between various species in the series. He decides to make up a simple lexicon of words that might be used by English speakers and Vulcans, using a different symbol to represent each word. For example, the word EARTH would be the symbol ", and VULCAN or VULCANS would be #. I or WE would be represented by the ! symbol, IS or AM or ARE would all be represented by =.

Here is a subset of a much larger collection of words that might be used to symbolize these different ideas.

```
! I/WE
# VULCAN/VULCANS
+ BIG
- SMALL
" EARTH
\ AND
= IS/AM/ARE
] HAS/HAVE
: MOON/MOONS
```

Using these words, a phrase like **`I AM VULCAN`** would be simply translated as **`!=#`**.
`VULCAN IS SMALL AND EARTH IS BIG` would be symbolized as **`#=-\"=+`**.

He needs your help translating simple English phrases into the symbols they represent in a simple "word for word" manner.

Input: Several one-character non alphanumeric symbols, each followed by one or more words that symbol represents. Any multiple words for each symbol are separated by the / character. Following the list of symbol/word combinations are a listing of sentences, in all uppercase, to be translated into symbols. **Note:** the judges data may have more symbol/word pairs than shown in the sample data.

Output: The symbolic representation of each sentence.

Sample input:

```
! I/WE
# VULCAN/VULCANS
+ BIG
- SMALL
" EARTH
\ AND
= IS/AM/ARE
] HAS/HAVE
: MOON/MOONS
I AM VULCAN
VULCAN IS SMALL AND EARTH IS BIG
VULCAN AND EARTH HAVE MOONS
```

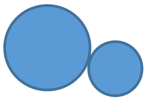
Sample output:

```
!=#
#=-\"=+
#\"]:
```

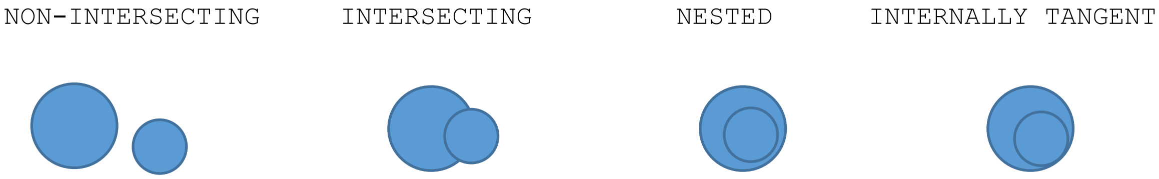
7. Konstantinos

Program Name: Konstantinos.java Input File: konstantinos.dat

In geometry class Konstantinos has been learning about circles being tangent (or not) in various situations. He knows that two circles are externally tangent when they look like this, intersecting at exactly one point, with no other points in common:



For two circles to be externally tangent, mathematically he knows that the sum of the radii for the two circles is equal to the distance between the two centers. He thinks he can figure out the math for other situations shown below but needs your help to write a program to do that. He remembers the distance formula from algebra class, which is: $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$



Input: Several data sets, each on one line consisting of six integer values x1, y1, r1, x2, y2, r2, representing the (x,y) circle center coordinates and corresponding radii for two circles. All values will have single space separation.

Output: For each data set, output the way the two circles intersect (or not), based on the five situations depicted by the diagrams shown above.

Sample input:

```
0 0 2 3 0 2
0 0 2 0 3 1
0 0 4 0 2 2
```

Sample output:

```
INTERSECTING
EXTERNALLY TANGENT
INTERNALLY TANGENT
```

8. Leonardo

Program Name: Leonardo.java Input File: leonardo.dat

According to the website Leonardo has found at URL (https://thoughtcatalog.com/nico-lang/2013/08/55-celebrities-whose-real-names-will-surprise-you/), the real names of several famous personalities are very interesting indeed. He decides to play around with these names, converting both stage and real names to initials, but with a twist. For the given name, he decided to make up initials from the last letter of each name part.

For example, **`Marilyn Monroe = Norma Jean Mortensen`** would convert to **`MM = ANN`**. Help him out by writing a program to do this simple but fun task.

Input: Several names of famous people, each on one line, first their personality name, followed by an equal sign, and then their real original given name. Either name will have one or more parts, but no extra designation like Jr. or II or anything else like that, just names.

Output: The initials for each name pair, first letters for the personality name, last letters for the real name, in the format demonstrated above and shown below.

Sample input:
```
Marilyn Monroe = Norma Jean Mortensen
Ben Kingsley = Krishna Pandit Bhanji
Katy Perry = Katy Hudson
Abigail Van Buren = Pauline Ester Friedman
Elvira = Cassandra Peterson
```

Sample output:
```
MM = ANN
BK = ATI
KP = YN
AVB = ERN
E = AN
```

9. Milo

Program Name: Milo.java Input File: milo.dat

Since neither the Cowboys or the Texans were in the Super Bowl this past February, Milo quickly became bored as he watched the game. Milo noticed that the logo for the game was Super Bowl LII. Roman numerals? Who uses those? Milo's grasp of the Roman numeral system is a little rusty, so he looked them up on Wikipedia. Here is some of what he found:

> The original pattern for Roman numerals used the symbols I, V, and X (1, 5, and 10) as simple tally marks. Each marker for 1 (I) added a unit value up to 5 (V), and was then added to (V) to make the numbers from 6 to 9: I, II, III, IIII, V, VI, VII, VIII, VIIII, X.
>
> The numerals for 4 (IIII) and 9 (VIIII) proved problematic and are generally replaced with IV (one less than 5) and IX (one less than 10). This feature of Roman numerals is called subtractive notation.
>
> The numbers from 1 to 10 are expressed in Roman numerals as follows:
>
> I, II, III, IV, V, VI, VII, VIII, IX, X.
>
> The system being basically decimal, tens and hundreds follow the same pattern:
>
> Thus 10 to 100 (counting in tens, with X taking the place of I, L taking the place of V and C taking the place of X):
>
> X, XX, XXX, XL, L, LX, LXX, LXXX, XC, C.
>
> Note that 40 (XL) and 90 (XC) follow the same subtractive pattern as 4 and 9.
>
> Similarly, 100 to 1000 (counting in hundreds):
>
> C, CC, CCC, CD, D, DC, DCC, DCCC, CM, M.
>
> Many numbers include hundreds, units and tens. The Roman numeral system being basically decimal, each "place" is added separately, in descending sequence from left to right, as with "arabic" numbers. For example, the number 39 is XXXIX, (three tens and a ten less one), 246 is CCXLVI (two hundreds, a fifty less ten, a five and a one). As each place has its own notation there is no need for place keeping zeros, so "missing places" can be simply omitted: thus 207, for instance, is written CCVII (two hundreds, a five and two ones) and 1066 becomes MLXVI (a thousand, a fifty and a ten, a five and a one)
>
> *Roman numerals. (2018, January 21). In Wikipedia, The Free Encyclopedia. Retrieved 21:06, February 20, 2018, from https://en.wikipedia.org/w/index.php?title=Roman_numerals&oldid=821541836*

In addition to the information from Wikipedia, Milo finds that to represent a number greater than 3999 requires using an over bar to indicate that a number must be multiplied by 1000. $\bar{V} = 5000$.

Milo needed to kill a little time until the half time show so he decided to whip up a little program to convert Roman numbers to Arabic numbers.

Input: A file that contains an unknown number of Roman numbers each listed on a separate line. All of the numbers will be greater than or equal to I and less than or equal to MMMCMXCIX. All of the Roman numbers in the file will be valid numbers as described above.

Output: The Arabic equivalent of each Roman numeral, each printed on a separate line.

Sample input:

```
IV
V
VII
XIX
XX
CV
MDCLXXVIII
```

Sample output:

```
4
5
7
19
20
105
1678
```

10. Oscar

Program Name: Oscar.java **Input File: oscar.dat**

Oscar works down at the sign company. His job is to make 24-inch wide signs with the phrases and words that are presented to him. Each sign has a 2-inch margin where nothing is printed except a border line. Each sign can have up to three lines of text, each letter or character 2 inches wide. The makeup of each sign must follow these rules:

1. A word cannot be divided.
2. Each line of words should be centered on the line.
3. If the spaces before and after a word or words on a line are unequal, place the larger number of spaces after the text.
4. If the sign has just one word, it should be placed on the middle line.
5. If the sign has two words, they should be placed in the first two lines.
6. If there are three or more words, all three lines must be used even if all the words will fit on fewer lines.
7. After at least one word has been placed on each line, lines should be filled with as many words as possible, starting with the top line.
8. If all the words will not fit on three lines the sign cannot be made.
9. If any of the words to be placed on the sign are too long to fit on a line, the sign cannot be made.
10. None of the text being considered will contain a period but may contain other symbols.

For the words, “Don’t Mess With Texas”, the first two words, “Don’t Mess” go on the first line, followed by “With” and “Texas” on the next two lines. It would be possible to have the words, “Mess With” on the second line, but this violates Rule 7 above.

Input: A file that contains an unknown number of lines each of which contains the text that should be placed on each sign.

Output: Each sign printed following the rules described above. Blank spaces between and around the text should be indicated by printing a period. The two-inch margin around the edge of the sign will be shown by “*” symbols. If the number of spaces on either side of the line of text is uneven, the larger number of spaces should be placed after the text. If a sign cannot be made, print **`SIGN CANNOT BE MADE`**. Each of the signs should be separated by one blank line.

Sample input:

```
Don't Mess With Texas
Get Your Crackerjacks Here
UIL Sign Company
Turn Right
```

Sample output:

```
************
*Don't.Mess*
*...With...*
*..Texas...*
************

SIGN CANNOT BE MADE

************
*...UIL....*
*...Sign...*
*.Company..*
************

************
*...Turn...*
*..Right...*
*..........*
************
```

11. Richa

Program Name: Richa.java Input File: richa.dat

The concept of GPA (grade point value) is new to Richa, a high school freshman this year, and she needs some help figuring out hers. The catalog of courses to be considered contains ten courses numbered from 140 to 149, and are designated either Honors or not, with three different possible credit lengths, a full year (4 credits), semester only (2 credits), or a quarter course (1 credit). To practice the calculation process on a simple level, she only wants to consider two courses at a time.

Grade point values are awarded as follows: A=4.5, B=3.5, C=2.5, D=1, and F=0. A '+' or a '-' can be added to each grade (except for F). This either adds or subtracts a quarter point from the value. Honors courses with a grade of C- or better earn an additional half point.

For example, a grade of A- for an Honors course would earn a grade point value of 4.75, subtracting a quarter point from 4.5, and then adding a half point. A D+ for a regular course would earn 1.25 points.

The total number of points earned for a course is the product of the grade point value earned and the number of credits earned. For example, the total points for an A- in a full year Honors course would be calculated as 4.75 times 4, which is a total of 19 grade points. A D+ in a semester course would earn a total of 1.25 times 2, or 2.5. The total GPA for these two courses would be the sum of the two grade point values, divided by the total number of credits, which is the calculation (19 + 2.5) / (4 + 2) = 21.5 / 6 = 3.583.

A C+ for course 144 and a B- for course 142 would result in a GPA of 3.417.
An A for 141 and a B for 148 earns a GPA of 4.0.

Input: An initial integer value N representing N course designations to follow, each consisting of a course number and a two character code, made up of one of the characters 'Y' of 'N' designating Honors or not, and then one of the three characters 'Y' for a full year course, 'S' for semester, and 'Q' for quarter. Following these N course data sets, several four-part data sets follow, each consisting of a letter grade A, B, C, D or F, possibly followed by a + or -, the course number, another letter grade, and then the second course. All data elements on a line are separated by single spaces.

Output: The resulting GPA of each pair of courses, formatted and rounded to three places of precision.

Sample input:
```
10
140 NS
141 NY
142 YY
143 YS
144 NS
145 YQ
146 NQ
147 YY
148 NY
149 YY
A- 149 D+ 143
C+ 144 B- 142
A 141 B 148
```

Sample output:
```
3.583
3.417
4.000
```

12. Romero

Program Name: Romero.java **Input File: romero.dat**

Mrs. Romero is the principal at Firethorne High School. Go Hotspurs! One of the many and varied duties she must perform is to sell and take tickets at the home football game on every other Friday night. Oddly enough, FHS changes the price for tickets for each game. For example, they charged $5.00 for adults and $3.00 for students and children not yet in school to attend the home opener. For the next game they charged $6.00 for adults but only $2.50 for students and children. Students always pay less than adults. At the end of each game Mrs. Romero knows the total attendance and the gross receipts for that game. She does not keep track of how many adults attended or how many students and children attended. At the end of the season the Firethorne ISD board of trustees decided that they would like to see the effect changing ticket prices had on attendance of each group of ticket buyers. They have requested a report showing the count of how many adults, students and children attended each game.

Luckily Mrs. Romero is a former algebra teacher and knows how to use a system of equations to calculate the attendance of each group. Here is how she did it.

At the first game total attendance was 3250. Gross receipts totaled $14,250. This provides enough information to come up with these two equations if she lets x be adults and y be students and children:

x+y=3250
5x+3y=14250

Now she multiplies the first equation by 5 to get:

5x+5y=16250
5x+3y=14250

Next she adds the opposite of the second equation to the first and gets:

2y=2000

Solves for y to get:

y=1000

Now she substitutes 1000 for y in the first equation:

x+1000=3250

Solves that for x and finds that:

x=2250

So, for the first game she can report that 2250 adults and 1000 students and children attended the game.

Mrs. Romero has a whole season’s worth of data stored in a file and doesn’t want to process all that data by hand. That is where you come in. Write a program that will process all her data and print a report of the date, total attendance, gross receipts, ticket prices, adult attendance and student/child attendance for each of the home football games that Firethorne High played this past season.

Input:

An initial value G representing the number of home games played this past season followed by G lines of data for each game. Each line of game data will contain a date shown as **mm/dd/yyyy** (all games were played in September, October, November or December), total attendance (stadium capacity is 4000), gross receipts for that game, ticket price for adults and ticket price for students and children. Neither ticket price was ever more than $9.50 and are never the same. All the data items on each line will have single space separation.

Output:

A report containing the data compiled for each home game in the exact format, column size and left alignment described here and shown in the examples. The report should begin with column headings for each data item: Date, Attendance, Gross, ATP, STP, Adults, Stu/Child. Each row should display the date as a long date (Month Day, Year), the gross receipts, adult ticket price and student ticket price using a dollar sign, comma separator if necessary, and two decimal places ($##,###.##) and display total, adult and students/children attendance as whole numbers. All the data should be left aligned within its column and there should be at least one space between each column. A final **pipe " | "** character is to end each line of the report, as shown.

12. Romero (continued)

No data or calculated values will exceed the column size requirements, and there will always be a minimum of one space separating the data elements in each column. For example, the Date column will contain dates no longer than 19 spaces, Attendance always starts in column 20, Gross in column 31 with values <$100K, etc.

Sample input:

```
2
9/8/2017 3250 14250 5.00 3.00
9/22/2017 2980 15027.50 6.00 2.50
```

Sample output:

```
Date               Attendance Gross      ATP   STP   Adults Stu/Child|
September 8, 2017  3250       $14,250.00 $5.00 $3.00 2250   1000     |
September 22, 2017 2980       $15,027.50 $6.00 $2.50 2165   815      |
```

UIL Computer Science Competition

State 2018

JUDGES PACKET - CONFIDENTIAL

I. Instructions

1. The attached printouts of the judge test data are provided for the reference of the contest director and programming judges. Additional copies may be made if needed for this purpose.

2. This packet must remain CONFIDENTIAL. Additional copies may be made and returned to schools when other confidential contest material is returned.

II. Table of Contents

Problem #1
60 Points

1. Ankur

Program Name: Ankur.java **Input File: None**

Test Output To Screen

```
3141.59
3141.592654
3.141593e+03
271.828
271.828183
2.718282e+02
```

Problem #2
60 Points

2. Catalina

Program Name: Catalina.java **Input File: catalina.dat**

Test Input File:

```
Absent WithOut Leave
Self-Contained Underwater Breathing Apparatus
Completely Automated Public Turing test to tell Computers and Humans Apart
Situation Normal, All Fouled Up
Junior Reserve Officer Training Corps
Missing In Action
Prisoner Of War
Special Weapons And Tactics
SEa Air and Land
All Points Bulletin
Cooperative for Assistance and Relief Everywhere
Diesel-Engined Road Vehicles
RAdio Detection And Ranging
SOund Navigation And Ranging
Subscriber Identification Module
Zone Improvement Plan
Fouled Up Beyond All Recognition
```

Test Output To Screen

```
AWOL
SCUBA
CAPTCHA
SNAFU
JROTC
MIA
POW
SWAT
SEAL
APB
CARE
DERV
RADAR
SONAR
SIM
ZIP
FUBAR
```

Problem #3
60 points

3. Dennis

Program Name: Dennis.java **Input File: dennis.dat**

Test Input File:

```
5
3
house
winter
cold
5
cat
turtle
aardvark
dog
fish
1
a
10
computer
science
java
perl
c++
123456789
stack
list
iterator
collection
6
phone
photo
pen
frame
stickynote
earbuds
```

Test Output To Screen

```
hwc
oio
unl
std
ee
 r
***
ctadf
auaoi
trrgs
 td h
 lv
 ea
  r
  k
*****
a
*
csjpc1slic
ocae+2tito
mivr+3asel
peal 4ctrl
un   5k ae
tc   6  tc
ee   7  ot
r    8  ri
     9   o
         n
**********
pppfse
hherta
oonair
nt mcb
eo eku
    yd
    ns
    o
    t
    e
******
```

Problem #4
60 points

4. Dinesh

Program Name: Dinesh.java **Input File: dinesh.dat**

Test Input File:

```
5
ONE TWO THREE FOUR
5
ONE TWO 4
ONE THREE 20
THREE FOUR 5
ONE FOUR 14
TWO FOUR 8
5
ONE TWO
ONE THREE
ONE FOUR
THREE FOUR
THREE TWO
ALPHA BETA GAMMA DELTA EPSILON
6
ALPHA BETA 1
ALPHA GAMMA 7
ALPHA EPSILON 3
BETA EPSILON 6
GAMMA EPSILON 2
EPSILON DELTA 3
3
ALPHA DELTA
BETA GAMMA
EPSILON BETA
A B C D E
5
A B 1
A C 1
A E 1
E D 1
B D 4
2
A D
B D
A B C D E
5
C D 3
A B 1
B E 7
C A 1
D E 1
2
D A
B E
A B C D E F
7
A B 1
B C 5
C D 3
D E 7
E F 7
A F 24
D A 10
3
C E
F A
D A
```

Test Output To Screen

```
ONE to TWO:4
ONE to THREE:17
ONE to FOUR:12
THREE to FOUR:5
THREE to TWO:13
ALPHA to DELTA:6
BETA to GAMMA:6
EPSILON to BETA:4
A to D:2
B to D:3
D to A:4
B to E:6
C to E:10
F to A:23
D to A:9
```

Problem #5
60 points

5. Emma

Program Name: Emma.java **Input File: emma.dat**

Test Input File:

```
3 4 5 7 9 8 10 1 2
```

Test Output To Screen

```
***   ***
***   ***
***   ***
   * *
    *
   * *
***   ***
***   ***
***   ***
==========
****    ****
****    ****
****    ****
****    ****
    *  *
     **
     **
    *  *
****    ****
****    ****
****    ****
****    ****
==========
*****     *****
*****     *****
*****     *****
*****     *****
*****     *****
     *   *
      * *
       *
      * *
     *   *
*****     *****
*****     *****
*****     *****
*****     *****
*****     *****
==========
*******       *******
*******       *******
*******       *******
*******       *******
*******       *******
*******       *******
*******       *******
       *     *
        *   *
         * *
          *
         * *
        *   *
       *     *
*******       *******
*******       *******
*******       *******
*******       *******
*******       *******
*******       *******
*******       *******
==========
*********         *********
*********         *********
*********         *********
*********         *********
*********         *********
*********         *********
*********         *********
*********         *********
*********         *********
         *       *
          *     *
           *   *
            * *
             *
            * *
           *   *
          *     *
         *       *
*********         *********
*********         *********
*********         *********
*********         *********
*********         *********
*********         *********
*********         *********
*********         *********
*********         *********
==========
********        ********
********        ********
********        ********
********        ********
********        ********
********        ********
********        ********
********        ********
        *      *
         *    *
          *  *
           **
           **
          *  *
         *    *
        *      *
********        ********
********        ********
********        ********
********        ********
********        ********
********        ********
********        ********
********        ********
==========
**********          **********
**********          **********
**********          **********
**********          **********
**********          **********
**********          **********
**********          **********
**********          **********
**********          **********
**********          **********
          *        *
           *      *
            *    *
             *  *
              **
              **
             *  *
            *    *
           *      *
          *        *
**********          **********
**********          **********
**********          **********
**********          **********
**********          **********
**********          **********
**********          **********
**********          **********
**********          **********
**********          **********
==========
* *
 *
* *
==========
**  **
**  **
  **
  **
**  **
**  **
==========
```

Problem #6
60 points

6. Johnny

Program Name: Johnny.java **Input File: johnny.dat**

Test Input File:

```
! I/WE
# VULCAN/VULCANS
+ BIG
- SMALL
" EARTH
\ AND
= IS/AM/ARE
: MOON/MOONS
& HE/THEY
$ SHE
% SAW
> WENT
@ TO
< THE
^ A
[ PLANET/PLANETS
] HAS/HAVE
; OR
I AM VULCAN
VULCAN IS SMALL AND EARTH IS BIG
VULCAN AND EARTH HAVE MOONS
I SAW EARTH
I AM A VULCAN
THEY ARE VULCANS
THEY WENT TO THE BIG PLANETS
SHE SAW A VULCAN MOON
THEY WENT TO VULCAN
VULCAN IS A SMALL PLANET
EARTH HAS A BIG MOON
EARTH IS BIG AND VULCAN IS SMALL
I AM VULCAN AND HE WENT TO EARTH
```

Test Output To Screen

```
!=#
#=-\"=+
#\"]:
!%"
!=^#
&=#
&>@<+[
$%^#:
&>@#
#=^-[
"]^+:
"=+\#=-
!=#\&>@"
```

Problem #7
60 points

7. Konstantinos

Program Name: Konstantinos.java **Input File: konstantinos.dat**

Test Input File:

```
0 0 2 3 0 2
0 0 2 0 3 1
0 0 4 0 2 2
1 1 1 2 2 3
1 2 3 4 5 6
2 1 5 11 13 10
1 1 1 4 2 2
```

Test Output To Screen

```
INTERSECTING
EXTERNALLY TANGENT
INTERNALLY TANGENT
NESTED
INTERSECTING
EXTERNALLY TANGENT
NON-INTERSECTING
```

Problem #8
60 points

8. Leonardo

Program Name: Leonardo.java **Input File: leonardo.dat**

Test Input File:

```
Marilyn Monroe = Norma Jean Mortensen
Ben Kingsley = Krishna Pandit Bhanji
Katy Perry = Katy Hudson
Abigail Van Buren = Pauline Ester Friedman
Elvira = Cassandra Peterson
Demi Moore = Demetria Guynes
Albert Brooks = Albert Einstein
Meg Ryan = Margaret Mary Emily Anne Hyra
Natalie Wood = Natalia Nikolaevna Zakharenko
Woody Allen = Allen Konigsberg
Joaquin Phoenix = Joaquin Rafael Bottom
Chevy Chase = Cornelius Crane Chase
Tina Fey = Elizabeth Stamatina Fey
Diane Keaton = Diane Hall
Michael Caine = Maurice Micklewhite
Larry King = Lawrence Harvey Zeigler
George Michael = Georgios Panayiotou
Whoopi Goldberg = Caryn Johnson
Hulk Hogan = Terry Jean Bollette
Bea Arthur = Bernice Frankel
Miranda July = Miranda Jennifer Grossinger
```

Test Output To Screen

```
MM = ANN
BK = ATI
KP = YN
AVB = ERN
E = AN
DM = AS
AB = TN
MR = TYYEA
NW = AAO
WA = NG
JP = NLM
CC = SEE
TF = HAY
DK = EL
MC = EE
LK = EYR
GM = SU
WG = NN
HH = YNE
BA = EL
MJ = ARR
```

Problem #9
60 points

9. Milo

Program Name: Milo.java **Input File: milo.dat**

Test Input File:

```
IV
V
VII
XIX
XX
CV
MDCLXXVIII
I
IX
XLI
XC
XCIX
CIV
CIX
CCXLV
CMXCIX
M
MIII
MXXXI
MCMXCIX
MMMCMXCIX
MMMCMXCVII
```

Test Output To Screen

```
4
5
7
19
20
105
1678
1
9
41
90
99
104
109
245
999
1000
1003
1031
1999
3999
3997
```

Problem #10
60 points

10. Oscar

Program Name: Oscar.java Input File: oscar.dat

Test Input File:

```
Don't Mess With Texas
Get Your Crackerjacks Here
UIL Sign Company
Turn Right
Word!
0123456789 0123456789
Watch For Ice On The Road
Dairy Queen
No Thru Traffic
Exit 163
This is too many words to be placed on a sign.
123456789101112 puppies
A B C D E F G H I
We Do Not Have Ice Cream
razzmatazz whizzbangs zigzagging
We Do Not Have Iceman Cream
A B C D E F G H I J K L M N O P
```

Test Output To Screen

```
************
*Don't.Mess*
*...With...*
*..Texas...*
************

SIGN CANNOT BE MADE

************
*...UIL....*
*...Sign...*
*.Company..*
************

************
*...Turn...*
*..Right...*
*..........*
************

************
*..........*
*..Word!...*
*..........*
************

************
*0123456789*
*0123456789*
*..........*
************

************
*Watch.For.*
*Ice.On.The*
*...Road...*
************

************
*..Dairy...*
*..Queen...*
*..........*
************

************
*....No....*
*...Thru...*
*.Traffic..*
************

************
*...Exit...*
*...163....*
*..........*
************

SIGN CANNOT BE MADE

SIGN CANNOT BE MADE

************
*A.B.C.D.E.*
*..F.G.H...*
*....I.....*
************

************
*We.Do.Not.*
*.Have.Ice.*
*..Cream...*
************

************
*razzmatazz*
*whizzbangs*
*zigzagging*
************

SIGN CANNOT BE MADE

SIGN CANNOT BE MADE
```

Problem #11
60 Points

11. Richa

Program Name: Richa.java **Input File: richa.dat**

Test Input File:

```
10
140 NS
141 NY
142 YY
143 YS
144 NS
145 YQ
146 NQ
147 YY
148 NY
149 YY
A- 149 D+ 143
C+ 144 B- 142
A 141 B 148
A- 141 B 148
A- 141 B- 148
F 145 D+ 147
C 144 B 148
B 144 C 148
F 146 F 140
A- 146 C+ 141
A 143 B+ 145
D+ 143 C- 147
F 140 D 149
A+ 142 A+ 147
```

Test Output To Screen

```
3.583
3.417
4.000
3.875
3.750
1.000
3.167
2.833
0.000
3.050
4.750
2.250
0.667
5.250
```

Problem #12
60 points

12. Romero

Program Name: Romero.java **Input File: romero.dat**

Test Input File:

```
7
9/8/2017 3250 14250 5.00 3.00
9/22/2017 2980 15027.50 6.00 2.50
10/6/2017 4000 21000 5.50 4.50
10/20/2017 3814 24190 7.00 5.00
11/3/2017 3963 21323.25 6.75 3.75
11/17/2017 3500 33250 9.50 9.45
12/2/2017 4000 14000 5.00 2.00
```

Test Output To Screen

```
Date               Attendance Gross      ATP   STP   Adults Stu/Child|
September 8, 2017  3250       $14,250.00 $5.00 $3.00 2250   1000     |
September 22, 2017 2980       $15,027.50 $6.00 $2.50 2165   815      |
October 6, 2017    4000       $21,000.00 $5.50 $4.50 3000   1000     |
October 20, 2017   3814       $24,190.00 $7.00 $5.00 2560   1254     |
November 3, 2017   3963       $21,323.25 $6.75 $3.75 2154   1809     |
November 17, 2017  3500       $33,250.00 $9.50 $9.45 3500   0        |
December 2, 2017   4000       $14,000.00 $5.00 $2.00 2000   2000     |
```