

# UIL COMPUTER SCIENCE WRITTEN TEST

# 2018 INVITATIONAL B

FEBRUARY/MARCH 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 INVITATIONAL B

**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.

Which of the following hexadecimal values is equivalent to  $11001000_2$ ?

- A) 51                      B) A8                      C) FF                      D) 64                      E) C8

## Question 2.

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

- A) 78      B) -93      C) 111      D) 213      E) -52

```
out.println(15+((19-11)*10)-17);
```

## Question 3.

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

- A) importstatic  
tpublicclass
- B) importstatic  
publicclass
- C) importstatic  
publicclass
- D) import  
static  
public  
class
- E) importstatic  
public  
class

```
out.print("import");
out.print("static\n\tpublic");
out.println("class");
```

## Question 4.

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

- A) andtigers
- B) lions and tigers
- C) lionsandt
- D) lionsandtigers
- E) lionstigers

```
String s="lions";
String t="tigers";
String u=s.concat("and"+t);
out.print(u);
```

## Question 5.

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

- A) true                      B) false

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

## Question 6.

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

- A) 3      B) 4      C) 5      D) 4.0
- E) There is no output. Will not compile due to a type mismatch.

```
int x;
double y=3.8712;
x=Math.floor(Math.round(y));
out.println(x);
```

## Question 7.

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

- A) 20      B) 19      C) 19.0      D) 19.35
- E) There is no output. Will not compile due to a type mismatch.

```
int w,x=8;
double y=4.25,z=7.60;
w=x+(int)(y+z);
out.print(w);
```

<p><b>Question 8.</b></p> <p>What is the output of the code segment to the right?</p> <p>A) penny B) nickel C) penny nickel dime quarter D) dime E) dime quarter</p>	<pre>String s="JAVA",t="java"; if(s.equals(t))     out.print("penny "); else if(s.equals("java"))     out.print("nickel "); else if(t.equalsIgnoreCase(s))     out.print("dime "); else if(t.equals("java"))     out.print("quarter");</pre>
<p><b>Question 9.</b></p> <p>What is the output of the code segment to the right?</p> <p>A) 1 4 7 9 B) 1 4 7 C) 1 2 3 4 5 6 7 8 9 D) 1 3 6 9 E) 4 7 9</p>	<pre>for(int i=1;i&lt;=9;i+=3)     out.print(i+" ");</pre>
<p><b>Question 10.</b></p> <p>What is the output of the code segment to the right?</p> <p>A) [4, 2, 0, 1, 4, 3] B) [0, 5, 0, 1, 4, 4] C) [2, 5, 2, 1, 4, 4] D) [2, 4, 2, 1, 4, 3] E) No output. Throws an ArrayIndexOutOfBoundsException .</p>	<pre>int[] list={2,5,0,1,4,3}; list[2]=list[0]; list[5]=list[2]; list[list[1]]=list[4]; out.print(Arrays.toString(list));</pre>
<p><b>Question 11.</b></p> <p>The code segment on the right is intended to read and print the values in datafile.dat all on one line. If datafile.dat contains the following values all on one line as shown here:</p> <p>5 9 1 7 3 4 6 2 3 8</p> <p>which of the following <u>cannot</u> replace &lt;code&gt; in the code segment? Assume that all necessary classes have been imported correctly and that the main method throws an IOException.</p> <p>A) s.next() B) s.nextLine() C) s.nextInt() D) s.nextDouble() E) All of the above <u>can</u> replace &lt;code&gt;.</p>	<pre>File f=new File("datafile.dat"); Scanner s=new Scanner(f); while(s.hasNext())     out.print(&lt;code&gt;+" ");</pre>
<p><b>Question 12.</b></p> <p>What is the output of the code segment to the right?</p> <p>A) 71 B) 61 C) 57 D) 70 E) 45</p>	<pre>int m=10,n=1,p=1; while(m&gt;0) {     p=p+n*m;     n++;     m-=2; } out.print(p);</pre>

<b>Question 13.</b>	
<p>In any given expression, which of the following operations would be done first?</p> <p><b>A) &lt;=</b>                      <b>B) type cast</b>                      <b>C) *</b>                      <b>D) ==</b>                      <b>E)  </b> (bitwise OR)</p>	
<b>Question 14.</b>	
<p>What is the output of the code segment to the right?</p> <p><b>A) 8</b>    <b>B) 16</b>    <b>C) 32</b>    <b>D) 64</b>    <b>E) 256</b></p>	<pre>out.print(Short.SIZE);</pre>
<b>Question 15.</b>	
<p>What is the output of the code segment to the right?</p> <p><b>A) [2, 0, 1]</b>  <b>B) [1, 2, 3]</b>  <b>C) [2, 1, 0]</b>  <b>D) [3, 2, 1]</b>  <b>E) [0, 1, 2]</b></p>	<pre>ArrayList&lt;Integer&gt; a=new ArrayList&lt;Integer&gt;(); a.add(3); a.add(2); a.add(1); a.add(0); a.remove(0); out.print(a);</pre>
<b>Question 16.</b>	
<p>Which of the following best describes the method shown on the right?</p> <p><b>A) Returns the sum of the values in the diagonal rows in array m.</b>  <b>B) Returns the largest value in array m.</b>  <b>C) Returns the sum of all of the values in array m.</b>  <b>D) Returns the sum of the largest value in each of the rows in array m.</b>  <b>E) Returns the smallest value in array m.</b></p>	<pre>public static int method(int[][] m) {     int a=0;     for(int x=0;x&lt;m.length;x++) {         int r=0;         for(int y=0;y&lt;m[x].length;y++)             if(m[x][y]&gt;r)                 r=m[x][y];         a+=r;     }     return a; }</pre>
<b>Question 17.</b>	
<p>What is the output of the code segment to the right?</p> <p><b>A) [3.14, 0.58, 2.72, 0.92, 1.2]</b>  <b>B) [0.58, 3.14, 2.72, 0.92, 1.2]</b>  <b>C) [3.14, 0.58, 1.2]</b>  <b>D) [3.14, 0.92, 1.2]</b>  <b>E) [3.14, 0.58, 0.92, 1.2]</b></p>	<pre>List&lt;Double&gt; list=new LinkedList&lt;Double&gt;(); list.add(3.14); list.add(2.72); list.add(0.92); list.set(1,0.58); list.add(1.20); list.get(2); out.print(list);</pre>
<b>Question 18.</b>	
<p>What is the output of the code segment shown on the right?</p> <p><b>A) 3</b>                      <b>B) 4</b>                      <b>C) 5</b>                      <b>D) 6</b>                      <b>E) 7</b></p>	<pre>String s="Anteaters hate aardvarks"; String[] t=s.split("a"); out.println(t.length);</pre>
<b>Question 19.</b>	
<p>What is the output of the code segment listed to the right?</p> <p><b>A) 0</b>                      <b>B) 1</b>                      <b>C) 90</b>                      <b>D) 91</b>  <b>E) There is no output due to an error.</b></p>	<pre>long e=90,f=91,g; g=f%2&gt;e/90?e%2:f/90; out.print(g);</pre>

**Question 20.**

Which of the following must replace **<code #1>** in the binary search method shown on the right?

- A)  $(\text{front} + \text{back}) / 2$
- B)  $\text{front} + \text{back}$
- C)  $(\text{back} - \text{front}) / 2$
- D)  $\text{back} / 2$
- E)  $\text{mid} / 2$

**Question 21.**

Which of the following must replace **<code #2>** in the binary search method shown on the right?

- A)  $\text{front}$
- B)  $\text{back}$
- C)  $\text{mid}$
- D) 1
- E) No additional code is required.

**Question 22.**

Assuming that **<code #1>** and **<code #2>** have been filled in correctly, what is the greatest possible number of comparisons that will be required within the `while` loop, if `list` contains 1024 elements?

- A) 10
- B) 1024
- C) 100
- D) 32
- E) 512

//Use the following code to answer questions 20, //21 and 22.

/\*list is a String array sorted in ascending order\*/

```
public static int search(String[] list,String s)
{
    int front=0,back=list.length-1;
    int mid=<code #1>;
    while(front<=back) {
        if(list[mid].equals(s))
            return <code #2>;
        else {
            if(list[mid].compareTo(s)<0)
                front=mid+1;
            else
                back=mid-1;
            mid=<code #1>;
        }
    }
    return -1;
}
```

**//Use the class shown here to answer questions 23 - 27.**

```
import java.util.ArrayList;

public class DataStructure<E> {

    private ArrayList<E> list=new ArrayList<E>();

    public int size() {
        return list.size();
    }

    public E next() {
        return list.get(size()-1);
    }

    public void add(E item) {
        list.add(item);
    }

    public E remove() {
        return list.remove(size()-1);
    }

    public boolean isEmpty() {
        return list.isEmpty();
    }

    public String toString() {
        return list.toString();
    }

}
```

<p><b>Question 23.</b></p> <p>Which of the following can replace <b>????</b> in the code segment shown on the right and ensure that the remainder of the code will compile and execute correctly?</p> <p>A) ArrayList B) Object C) String D) A, B, and C E) B and C</p>	
<p><b>Question 24.</b></p> <p>What is printed by <b>line #1</b> in the code segment to the right?</p> <p>A) snow B) hail C) rain D) dust E) wind</p>	<pre> DataStructure&lt;????&gt; ds=new DataStructure&lt;????&gt;(); ds.add("rain"); ds.add("snow"); ds.add("wind"); ds.add("hail"); ds.add("dust"); ds.remove(); out.println(ds.next());<b>//line #1</b> ds.add("sleet"); <b>//comment #1</b> while(!ds.isEmpty())     out.print(ds.remove()+" "); <b>//comment #2</b> out.println("\n"+ds.size());<b>//line #2</b> </pre>
<p><b>Question 25.</b></p> <p>What is the output of the while loop shown between <b>comment #1</b> and <b>#2</b> in the code segment to the right?</p> <p>A) sleet rain snow wind B) sleet wind snow rain C) sleet hail wind snow rain D) rain snow wind hail sleet E) rain snow wind sleet</p>	
<p><b>Question 26.</b></p> <p>What is printed by <b>line #2</b> in the code segment to the right?</p> <p>A) 0 B) 1 C) 2 D) 3 E) 4</p>	
<p><b>Question 27.</b></p> <p>Which of the following data structures does the class DataStructure, shown above, implement?</p> <p>A) set      B) linked list      C) priority queue      D) stack      E) queue</p>	
<p><b>Question 28.</b></p> <p>Which of the following is not a valid identifier?</p> <p>A) List      B) total_weight      C) FINAL      D) public      E) closed</p>	



**Question 29.**

Which of the following must replace **<code>** in the class Student shown on the right so that the code segment shown between the comments will function as a constructor?

- A) String
- B) void
- C) int
- D) static
- E) No additional code is required.

```
public class Student {

    public String get_name() {
        return name;
    }

    public double average() {
        return total/numGrades;
    }

    public void add_grade(int g) {
        total+=g;
        numGrades++;
    }
    //comment
    public <code> Student(String name) {
        this.name=name;
    }
    //comment

    private String name;
    private int numGrades;
    private double total;
}
```

**Question 30.**

Assuming that **<code>** has been filled in correctly, which of the following, if present in client code, will instantiate a Student object for a student named Larry?

- A) Student s1=new Student("Larry");
- B) Student s1="Larry";
- C) Student.name="Larry";
- D) Student s1=get\_name("Larry");
- E) name=get\_name("Larry");

**Question 31.**

Assuming s1 and s2 are both Student objects constructed with the name "Larry", what is printed by this client code?

```
s1.add_grade(95);s1.add_grade(85);
s2.add_grade(95);s2.add_grade(85);
out.println(s1==s2);
```

- A) true      B) false      C) 0      D) 1
- E) s1 and s2 cannot be compared because the Student class does not contain a compareTo method.

**Question 32.**

What is printed by the code segment shown on the right?

- A) 12 2 6
- B) 7 0 10
- C) 10 1 7
- D) 7 1 8
- E) 12 1 6

```
int x=25,y=12,z=2;
while(y>1) {
    x-=Math.min(y, z);
    y=x/z;
    z++;
}
out.println(x+" "+y+" "+z);
```

**Question 33.**

The output shown on the right was produced by printing the contents of an array each time a particular sort made a pass over that array. Which of the following sort routines was it?

- A) quick sort
- B) merge sort
- C) bubble sort
- D) selection sort
- E) insertion sort

```
[5, 3, 1, 0, 2, 4] Original List
[0, 3, 1, 5, 2, 4]
[0, 1, 3, 5, 2, 4]
[0, 1, 2, 5, 3, 4]
[0, 1, 2, 3, 5, 4]
[0, 1, 2, 3, 4, 5]
[0, 1, 2, 3, 4, 5] Sorted List
```

**Question 34.**

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

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

```
byte i=2,j=3,k=4;
out.print(i<<j>>k);
```

**Question 35.**

To use the `Collections.sort()` method all elements within the collection to be sorted must implement the \_\_\_\_\_ interface?

- A) Arrays  
B) Collections  
C) ArrayList  
D) List  
E) Comparable

**Question 36.**

Which of the following Boolean expressions will produce the truth table shown on the right?

- A)  $A * B + A$   
B)  $A * B \oplus A$   
C)  $A + B \oplus A$   
D)  $A \oplus B \oplus A$   
E)  $A + B * A$

A	B	
T	T	T
T	F	T
F	T	T
F	F	F

**Question 37.**

Each element within a linked list data structure is stored within a \_\_\_\_\_?

- A) array    B) edge    C) identifier    D) class    E) node

**Question 38.**

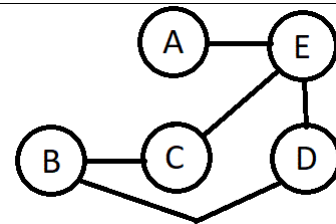
Which of the following is the correct infix representation of the postfix expression shown on the right? (The operands are 15, 4, 2, 3, and 5)

- A)  $15-4*2/(3+5)$   
B)  $15-4*2/3+5$   
C)  $15*4+2/3-5$   
D)  $2*3+4/5-15$   
E)  $(3+5)/4*2-15$

15 4 2 \* 3 5 + / -

**Question 39.**

How many pairs of vertices in the graph shown on the right are adjacent?

**Question 40.**

What is the decimal equivalent of the 8 bit binary two's complement value 10111000?

# UIL COMPUTER SCIENCE WRITTEN TEST

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

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

## FOR ADMINISTRATIVE USE ONLY

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

	Score	Initials
Judge #1:	<input type="text"/>	<input type="text"/>
Judge #2:	<input type="text"/>	<input type="text"/>
Judge #3:	<input type="text"/>	<input type="text"/>

# ★ ANSWER KEY – CONFIDENTIAL ★

## UIL COMPUTER SCIENCE – 2018 INVITATIONAL B

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

- |                  |                  |                  |                   |
|------------------|------------------|------------------|-------------------|
| 1) <u>  E  </u>  | 11) <u>  E  </u> | 21) <u>  C  </u> | 31) <u>  B  </u>  |
| 2) <u>  A  </u>  | 12) <u>  A  </u> | 22) <u>  A  </u> | 32) <u>  C  </u>  |
| 3) <u>  C  </u>  | 13) <u>  B  </u> | 23) <u>  E  </u> | 33) <u>  D  </u>  |
| 4) <u>  D  </u>  | 14) <u>  B  </u> | 24) <u>  B  </u> | 34) <u>  B  </u>  |
| 5) <u>  A  </u>  | 15) <u>  C  </u> | 25) <u>  C  </u> | 35) <u>  E  </u>  |
| 6) <u>  E  </u>  | 16) <u>  D  </u> | 26) <u>  A  </u> | 36) <u>  C  </u>  |
| 7) <u>  B  </u>  | 17) <u>  E  </u> | 27) <u>  D  </u> | 37) <u>  E  </u>  |
| 8) <u>  D  </u>  | 18) <u>  D  </u> | 28) <u>  D  </u> | 38) <u>  A  </u>  |
| 9) <u>  B  </u>  | 19) <u>  B  </u> | 29) <u>  E  </u> | *39) <u>  5  </u> |
| 10) <u>  C  </u> | 20) <u>  A  </u> | 30) <u>  A  </u> | *40) <u> -72 </u> |

\* 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:

1.	E	Cut the binary value into two four bit pieces. 1100 and 1000. Then convert those to the hexadecimal digits of the new number. $1100_2=12_{10}=C_{16}$ . $1000_2=8_{10}=8_{16}$ . Therefore $11001000_2 = C8_{16}$ .																																		
2.	A	$15+((19-11)*10)-17=$ $15+(8*10)-17=$ $15+80-17=$ $95-17=$ $78$																																		
3.	C	The <code>print</code> method does not advance the cursor to the next line. <code>\n</code> is the escape sequence for a new line and <code>\t</code> is the escape sequence for a tab																																		
4.	D	The <code>concat</code> method concatenates (adds) its arguments to the string <code>s</code> . There are no spaces in the original strings so none are added by <code>concat</code> .																																		
5.	A	<code>!(true&amp;&amp;!(false  true))=</code> <code>!(true&amp;&amp;true)=</code> <code>!(true&amp;&amp;false)=</code> <code>!false=</code> <code>true</code>																																		
6.	E	<code>round</code> returns a long. <code>floor</code> requires a double argument and returns a double which cannot be assigned to an <code>int</code> variable.																																		
7.	B	$8+(\text{int})(4.25+7.60)=$ $8+(\text{int})11.85=$ $8+11=$ //Casting truncates the decimal portion of the value. $19$																																		
8.	D	Java is case sensitive so the first two conditions are false. Once a condition is true, the remainder of the if/else statement is skipped.																																		
9.	B	$1 + 3 = 4$ $4 + 3 = 7$ $7 + 3 = 10$ Stop before entering the loop body again.																																		
10.	C	<table><tr><td>original list</td><td>2</td><td>5</td><td>0</td><td>1</td><td>4</td><td>3</td></tr><tr><td><code>list[2]=list[0]</code></td><td>2</td><td>5</td><td>2</td><td>1</td><td>4</td><td>3</td></tr><tr><td><code>list[5]=list[2]</code></td><td>2</td><td>5</td><td>2</td><td>1</td><td>4</td><td>2</td></tr><tr><td><code>list[list[1]]=list[4]</code></td><td>2</td><td>5</td><td>2</td><td>1</td><td>4</td><td>4</td></tr></table>							original list	2	5	0	1	4	3	<code>list[2]=list[0]</code>	2	5	2	1	4	3	<code>list[5]=list[2]</code>	2	5	2	1	4	2	<code>list[list[1]]=list[4]</code>	2	5	2	1	4	4
original list	2	5	0	1	4	3																														
<code>list[2]=list[0]</code>	2	5	2	1	4	3																														
<code>list[5]=list[2]</code>	2	5	2	1	4	2																														
<code>list[list[1]]=list[4]</code>	2	5	2	1	4	4																														
11.	E	All of the tokens in <code>datafile.dat</code> can be interpreted as strings, integers or real numbers (double).																																		
12.	A	<table><tr><td>m</td><td>n</td><td>p</td></tr><tr><td>10</td><td>1</td><td>1</td></tr><tr><td>8</td><td>2</td><td>11</td></tr><tr><td>6</td><td>3</td><td>27</td></tr><tr><td>4</td><td>4</td><td>45</td></tr><tr><td>2</td><td>5</td><td>61</td></tr><tr><td>0</td><td>6</td><td>71</td></tr></table>							m	n	p	10	1	1	8	2	11	6	3	27	4	4	45	2	5	61	0	6	71							
m	n	p																																		
10	1	1																																		
8	2	11																																		
6	3	27																																		
4	4	45																																		
2	5	61																																		
0	6	71																																		
13.	B	type cast then <code>*</code> then <code>&lt;=</code> then <code>==</code> then <code> </code> (bitwise OR)																																		
14.	B	The <code>size</code> method returns the memory required to store a value of this type.																																		
15.	C	The argument for <code>remove</code> refers to the index value. Therefore the 3 is removed from the list.																																		
16.	D	The outer loop steps through each row. The inner loop traverses each row searching for the largest value and placing it into variable <code>r</code> . When the inner loop stops, <code>r</code> is added to variable <code>a</code> . When the outer loop stops, <code>a</code> (the sum) is returned.																																		
17.	E	<code>set</code> replaces the current element with the new one at the designated index value. <code>get</code> returns but does not remove the element from the list.																																		
18.	D	The delimiter "a" is removed from the string. Where the delimiter occurs in succession, an empty string is placed in the array. Here is the contents of the array <code>t</code> : <table><tr><td>Ante</td><td>ters h</td><td>te</td><td>empty string</td><td>rdv</td><td>rks</td></tr></table>							Ante	ters h	te	empty string	rdv	rks																						
Ante	ters h	te	empty string	rdv	rks																															

19.	B	The statement is equivalent to: <pre>if (f%2&gt;e/90)     g=e%2; else     g=f/90;</pre>																						
20.	A	mid represents the middle element of each segment of the array to be searched.																						
21.	C	At this point the element being searched for has been found at list[mid].																						
22.	A	Run time efficiency of a binary search is O(log n). More specifically, log <sub>2</sub> n. Because the list is cut in half with each pass. log <sub>2</sub> n=1024 which means n=10 because 2 <sup>10</sup> =1024. Another way to look at it is two see how many times 1024 can be divided by 2. <table><tr><td>1024</td><td></td></tr><tr><td>512</td><td>1</td></tr><tr><td>256</td><td>2</td></tr><tr><td>128</td><td>3</td></tr><tr><td>64</td><td>4</td></tr><tr><td>32</td><td>5</td></tr><tr><td>16</td><td>6</td></tr><tr><td>8</td><td>7</td></tr><tr><td>4</td><td>8</td></tr><tr><td>2</td><td>9</td></tr><tr><td>1</td><td>10</td></tr></table>	1024		512	1	256	2	128	3	64	4	32	5	16	6	8	7	4	8	2	9	1	10
1024																								
512	1																							
256	2																							
128	3																							
64	4																							
32	5																							
16	6																							
8	7																							
4	8																							
2	9																							
1	10																							
23.	E	If ????? is replaced with ArrayList and the java.util package is imported the line with the missing code will compile, however, the lines that follow that contain the add method will not compile because they have a string argument.																						
24.	B	ds is a stack, which means the last element placed into ds will be the first one to come out. The call to remove pops “dust” off of the stack. next returns, but does not remove, the item that is on top of the stack which in this case is “hail”																						
25.	C	Since ds is a stack, elements are removed in a first in, last out fashion.																						
26.	A	isEmpty returns true when no elements are left in ds.																						
27.	D	See #24 and #25.																						
28.	D	public is a reserved word (keyword).																						
29.	E	Constructor methods do not have a return type. Answers A, C, and D will not compile. Answer choice B will compile if used but the method will no longer serve as a constructor if it has a return type.																						
30.	A	To instantiate an object there must be a call to a constructor. Choice A is the only answer choice that includes a call to the Student class constructor.																						
31.	B	s1 and s2 refer to two different objects even though each object’s instance variables contain the same values.																						
32.	C	<table><tr><td>x</td><td>y</td><td>z</td></tr><tr><td>25</td><td>12</td><td>2</td></tr><tr><td>23</td><td>11</td><td>3</td></tr><tr><td>20</td><td>6</td><td>4</td></tr><tr><td>16</td><td>4</td><td>5</td></tr><tr><td>12</td><td>2</td><td>6</td></tr><tr><td>10</td><td>1</td><td>7</td></tr></table>	x	y	z	25	12	2	23	11	3	20	6	4	16	4	5	12	2	6	10	1	7	
x	y	z																						
25	12	2																						
23	11	3																						
20	6	4																						
16	4	5																						
12	2	6																						
10	1	7																						
33.	D	During each pass of a selection sort the smallest element (in this case) is selected from the list and swapped with the first element. Then the index number of the first element is incremented to account for those elements that have already been placed in order.																						
34.	B	Bit shift operators are done left to right. 2<<3>>4= (2*2 <sup>3</sup> )/2 <sup>4</sup> = 2*8/16= 16/16= 1																						
35.	E	The sort method within the Collections class requires that the collection to be sorted contain objects that are derived from a class that implements the Comparable interface and subsequently have a compareTo method.																						

36.	C	<table> <tr><td>A</td><td>B</td><td><math>A*B+A</math></td></tr> <tr><td>T</td><td>T</td><td>T</td></tr> <tr><td>T</td><td>F</td><td>T</td></tr> <tr><td>F</td><td>T</td><td>F</td></tr> <tr><td>F</td><td>F</td><td>F</td></tr> </table> <table> <tr><td>A</td><td>B</td><td><math>A*B\oplus A</math></td></tr> <tr><td>T</td><td>T</td><td>F</td></tr> <tr><td>T</td><td>F</td><td>T</td></tr> <tr><td>F</td><td>T</td><td>F</td></tr> <tr><td>F</td><td>F</td><td>F</td></tr> </table> <table> <tr><td>A</td><td>B</td><td><math>A+B\oplus A</math></td></tr> <tr><td>T</td><td>T</td><td>T</td></tr> <tr><td>T</td><td>F</td><td>T</td></tr> <tr><td>F</td><td>T</td><td>T</td></tr> <tr><td>F</td><td>F</td><td>F</td></tr> </table> <table> <tr><td>A</td><td>B</td><td><math>A\oplus B\oplus A</math></td></tr> <tr><td>T</td><td>T</td><td>T</td></tr> <tr><td>T</td><td>F</td><td>F</td></tr> <tr><td>F</td><td>T</td><td>T</td></tr> <tr><td>F</td><td>F</td><td>F</td></tr> </table> <table> <tr><td>A</td><td>B</td><td><math>A+B*A</math></td></tr> <tr><td>T</td><td>T</td><td>T</td></tr> <tr><td>T</td><td>F</td><td>T</td></tr> <tr><td>F</td><td>T</td><td>F</td></tr> <tr><td>F</td><td>F</td><td>F</td></tr> </table>	A	B	$A*B+A$	T	T	T	T	F	T	F	T	F	F	F	F	A	B	$A*B\oplus A$	T	T	F	T	F	T	F	T	F	F	F	F	A	B	$A+B\oplus A$	T	T	T	T	F	T	F	T	T	F	F	F	A	B	$A\oplus B\oplus A$	T	T	T	T	F	F	F	T	T	F	F	F	A	B	$A+B*A$	T	T	T	T	F	T	F	T	F	F	F	F
A	B	$A*B+A$																																																																											
T	T	T																																																																											
T	F	T																																																																											
F	T	F																																																																											
F	F	F																																																																											
A	B	$A*B\oplus A$																																																																											
T	T	F																																																																											
T	F	T																																																																											
F	T	F																																																																											
F	F	F																																																																											
A	B	$A+B\oplus A$																																																																											
T	T	T																																																																											
T	F	T																																																																											
F	T	T																																																																											
F	F	F																																																																											
A	B	$A\oplus B\oplus A$																																																																											
T	T	T																																																																											
T	F	F																																																																											
F	T	T																																																																											
F	F	F																																																																											
A	B	$A+B*A$																																																																											
T	T	T																																																																											
T	F	T																																																																											
F	T	F																																																																											
F	F	F																																																																											
37.	E	Each node contains the element and a reference variable that points to the next node in the list.																																																																											
38.	A	$15 \ 4 \ 2 * 3 \ 5 + / -$ $15 \ (4*2) \ (3+5) / -$ $15 \ ((4*2) / (3+5)) -$ $15 - 4*2 / (3+5)$																																																																											
39.	5	A pair of vertices are adjacent if they are connected by an edge.																																																																											
40.	-72	We know the decimal value is negative since the left most bit is one. Start by taking the complement (flip the bits) which gets 01000111. Then add one to get 01001000. Convert to decimal to get 72 and we know it is negative so the answer is -72.																																																																											



# Computer Science Competition Invitational B 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	Baldo
Problem 2	Barb
Problem 3	Constanza
Problem 4	Dilbert
Problem 5	Emily
Problem 6	Fa
Problem 7	Judith
Problem 8	Luann
Problem 9	Lucy
Problem 10	Micaela
Problem 11	Nikita
Problem 12	Peter



# 1. Baldo

**Program Name:** Baldo.java

**Input File:** none

You and your classmate Baldo are enrolled in Computer Science 1 at Imaginary High School. Go Dragons! Your teacher, Mr. Snigglefritz, is requiring that every CS 1 student memorize the ASCII value associated with each of the letters of the alphabet both lower and uppercase. Baldo has decided to type up a chart that contains the letters and their ASCII values so that he can study without being near his computer or phone. You on the other hand see a programming opportunity.

**Input:** None.

**Output:** A chart of all 26 English alphabet letters and their ASCII values, in natural order. On each row there should be four items listed, with single space separation:

- an uppercase letter
- that letter's ASCII value
- the same letter in lowercase
- the ASCII value for the lowercase letter.

**Sample input:**

None

**Sample output:**

```
A 65 a 97
B 66 b 98
C 67 c 99
D 68 d 100
E 69 e 101
F 70 f 102
G 71 g 103
H 72 h 104
I 73 i 105
J 74 j 106
K 75 k 107
L 76 l 108
M 77 m 109
N 78 n 110
O 79 o 111
P 80 p 112
Q 81 q 113
R 82 r 114
S 83 s 115
T 84 t 116
U 85 u 117
V 86 v 118
W 87 w 119
X 88 x 120
Y 89 y 121
Z 90 z 122
```

## 2. Barb

**Program Name:** Barb.java

**Input File:** barb.dat

Barb has just learned some basic string manipulation techniques and is fiddling around with them, taking any word that comes to mind and messing with it.

Study the output below and reproduce it, given a single lowercased input word each time.

**Input:** Several lowercased single words, each on one line, containing only alpha characters.

**Output:** The original word, plus three more variations as shown in the sample output below.

**Sample input:**

```
red
yellow
green
```

**Sample output:**

```
red DER er ED
yellow WOLLEY ley LOW
green NEERG erg EEN
```

### 3. Constanza

**Program Name:** Constanza.java

**Input File:** constanza.dat

Constanza loves art, especially using the seven colors of the rainbow, indicated by the acronym, ROYGBIV (*red, orange, yellow, green, blue, indigo, violet*). She has collected numerous pieces of abstract “rainbow” art, and has scanned each image into a file that contains a listing of the rainbow colors of each pixel. For one of her art pieces, the first part of the listing looks like this (*shown here in three columns, but all in one column in the actual listing*), and continues for many more lines:

yellow	green	green
orange	blue	red
green	yellow	violet
indigo	red	yellow
green	blue	indigo
blue	blue	red
orange	indigo	red

She wants to analyze the pixel frequency of each color for this art piece, creating a horizontal bar graph that shows this, a display that looks like the one shown below, with the rainbow colors listed in order, each followed by the number of stars indicating the frequency of that color in the art piece.

```
red      ****
orange  **
yellow  ***
green   ****
blue    ****
indigo  ***
violet  *
```

**Input:** A vertical, single column listing of only rainbow colors (red, orange, yellow, green, blue, indigo, violet) in the format as shown above, except with each on one line, and no multiple columns. The file listing is for one and only one art piece. There will be at least one instance of each color.

**Output:** A single horizontal bar graph display representing the frequency of the colors in the data file. Both the color labels and bars of stars must be left-aligned, and spaced exactly as shown above.

**Sample input and output shown in example above.**

## 4. Dilbert

**Program Name:** Dilbert.java

**Input File:** dilbert.dat

Dilbert is the chairman of Firethorne High School Student Council's fundraising committee. The committee's October fund raising activity this year was to sell Halloween candy for customers to pass out on Halloween night to trick or treaters. Dilbert's record keeping for the fundraiser consists of a list of names, dates and dollar amounts turned in each day by various committee members. Dilbert promised a prize to the top seller for the fundraiser, so he needs to know which of his committee members did the best job of selling candy. Since you serve on the technology committee of the Student Council, Dilbert has asked you to write a program that will process his list of names, dates and sales, producing a report that will help him determine the top prize winner.

**Input:** Several sets of data lines, each containing a first name, last name, date, and dollar amount turned in on that day, with single space separation between each item. All dates will be listed in the format **mm/dd/yyyy**. Dollar amounts range in value from at least one dollar to hundreds of dollars, do not contain a dollar sign (\$) and will be listed with two decimal places of precision. Each committee member is listed by first and last name. No two members have the same name.

**Output:** A list that contains an entry for each committee member where each person's name is on the first line followed on subsequent lines by each date money was turned in by that person, with the amount in dollar format immediately following the date, separated by a single space. The entries for each member are to be sorted by date, earliest to latest. The final line for each committee member should be a line that contains the total sales for that committee member. The list of committee member records should be sorted in **descending** order based on total sales, with the first name listed in the report having the greatest total sales. All dates should be displayed in the **mm/dd/yyyy** format, and all dollar amounts should be preceded by a dollar sign (\$), with no space between it and the first digit of the dollar value. Here is a template for each committee member's output:

```
FirstName LastName
10/1/2017 $##.##
10/15/2017 $##.##
10/20/2017 $##.##
10/29/2017 $##.##
Total Sales $##.##
```

Each committee member's record should be followed by a blank line. The last line of the output should be the grand total for all sales during the fundraiser, shown as follows:

**Grand Total \$###.##**

**Sample input:**

```
Cory Randolph 10/6/2017 25.25
Idalia Saucedo 10/3/2017 15.00
Alonzo Carrasco 10/6/2017 31.10
Steve Flanary 10/2/2017 20.00
Cory Randolph 10/30/2017 18.50
Steve Flanary 10/15/2017 12.75
Cory Randolph 10/14/2017 35.90
Idalia Saucedo 10/20/2017 26.35
```

Total Sales \$79.65

```
Idalia Saucedo
10/3/2017 $15.00
10/20/2017 $26.35
Total Sales $41.35
```

```
Steve Flanary
10/2/2017 $20.00
10/15/2017 $12.75
Total Sales $32.75
```

**Sample output:**

```
Cory Randolph
10/6/2017 $25.25
10/14/2017 $35.90
10/30/2017 $18.50
```

```
Alonzo Carrasco
10/6/2017 $31.10
Total Sales $31.10
```

Grand Total \$184.85

## 5. Emily

**Program Name: Emily.java**

**Input File: emily.dat**

In geometry, the class is reviewing area formulas of standard geometric figures, and Emily has decided to write a program to double check her work, especially with trapezoids. She knows that the area of a trapezoid is the **average of the bases times the height**, but wants to make sure she is doing the work correctly.

**Input:** Several lines of data, each line containing three integers representing the two bases and height of a trapezoid.

**Output:** The area of the trapezoid, with decimal precision of one place if the value is fractional, otherwise no decimal point showing for whole number results.

**Sample input:**

```
5 7 3
8 3 12
3 4 5
```

**Sample output:**

```
18
66
17.5
```

## 6. Fa

**Program Name:** Fa.java

**Input File:** fa.dat

In a recent thought experiment Fa conducted dealing with balancing lists of values, he imagined a list of random numbers, all non-negative within a limited range (each value less than 100). He stored them in original order in a contiguous data structure, and wanted to find the balancing point, or fulcrum, of the list. He realized that this point could either be in between two values of the list, or at an exact position of one of those values in the list, depending on the sum of the values before and after the fulcrum. He wanted the “before” and “after” sums to be equal, or as close as possible to being equal, so that the list was balanced at that point.

For example, in a list containing the values, 1 2 3 2 1, the fulcrum clearly is at position 2, right at the value 3, since the sum of the values before and after the fulcrum are equal, a sum of 3 on each side, making a perfectly balanced list.

1 2 3 2 1  
          ^

He decided to indicate his findings for this list showing the “before” sum, the position of the fulcrum, and then the “after” sum, like this: 3 ^2 3

For another list he tried, 3 6 3 3 5 8, he found the balancing point to be at position 3, with a sum of 12 (3+6+3) before that position, and 13 (5+8) after. He realized this wasn’t a perfect balance, but he was content with “almost”, since it was the best positioning of a fulcrum to best balance the list.

3 6 3 3 5 8  
          ^

The report for this example was: 12 ^3 13

In a third list of values, 3 3 3 3 4 3, the fulcrum was actually in between positions 2 and 3, with a sum before of 9, and after of 10.

3 3 3 3 4 3  
          ^

Since the fulcrum ended up between two elements, the report format was slightly different, with the two index positions listed before and after the caret symbol, like this: 9 2^3 10

**Input:** Several data sets, each consisting of two lines. The first line contains an integer N ( $2 \leq N \leq 50$ ) indicating the number of elements in the list, followed on the next line by N random non-negative integers, no greater in value than 100.

**Output:** For each data set, show the final report formatted as shown above, with the “before” sum listed first, the fulcrum next, followed by the “after” sum, with single space separation. *Note: it is guaranteed there will be a “unique” fulcrum for each data set.*

**Sample input:**

```
5
1 2 3 2 1
6
3 6 3 3 5 8
6
3 3 3 3 4 3
```

**Sample output:**

```
3 ^2 3
12 ^3 13
9 2^3 10
```

## 7. Judith

**Program Name:** Judith.java

**Input File:** judith.dat

Judith is doing research using some very precise floating-point measurement values, some very large, some very small, and her first step is simply to sort them in ascending order. Can you help her do that?

The sample input and output below show an example of what she is dealing with.

**Input:** A datafile containing several non-negative floating-point values.

**Output:** An ascending sorted list of the values, each value shown in the **exact format** as was provided in the data file.

**Sample input:**

```
395.96774839412213864694087795
375.286439548498
8.412190040221
0.382560882678260591251068
0.23943358855017051969126499
95.96038946102548087733835
0.392863116206040
557.91740143772795279
46.8020528027329
0.91986136436504
```

**Sample output:**

```
0.23943358855017051969126499
0.382560882678260591251068
0.392863116206040
0.91986136436504
8.412190040221
46.8020528027329
95.96038946102548087733835
375.286439548498
395.96774839412213864694087795
557.91740143772795279
```

## 8. Luann

**Program Name:** Luann.java

**Input File:** luann.dat

This past summer Luann collected data on how many turtles she saw in the pond behind her home. Several days in a row she would go down to the creek and count the turtles in the pond and then record her findings in a data file. Luann didn't count turtles every single day. She would go on consecutive days for a while then skip a few days. She never counted turtles more than nine days in a row. Now that summer is over, Luann would like to chart her findings. She wants a different chart for each set of consecutive days that she counted turtles. Let's help her write a program that will process the data and print out the charts she wants.

**Input:** A single value C representing the number of charts to be printed. N will be followed by C sets of data each beginning with a number N that represents the number of coordinate pairs to follow. Each coordinate pair will be on a separate line where the first value is x and the second value y. Both x and y will be greater than or equal to 1 and less than or equal to 9. The x values will always begin with one and continue in sequential order with no skips. For example, if there are 7 coordinate pairs, the x values will always be 1 – 7 without any skips.

**Output:** A chart for each set of data. Each chart should have an x and y axis labeled starting at one (1). The y axis should be formed using pipes (|) and the x axis should be formed using underscores (\_). The y axis and its labels should take up two spaces in each row and the x axis and its labels should take up two rows. Each data point (y value) should be marked with a capital X. Any space within the chart that does not have an axis, a label or a data point must be filled with a blank. Each chart should be followed by the line, "=====".

**Sample input:**

```
2
7
1 1
2 2
3 3
4 4
5 5
6 3
7 6
9
1 1
2 4
3 9
4 9
5 9
6 7
7 5
8 2
9 1
```

**Sample output:**

```
6 |      X
5 |      X
4 |      X
3 |  X   X
2 |  X
1 |X

      1234567
=====
9 |   XXX
8 |
7 |      X
6 |
5 |      X
4 |  X
3 |
2 |      X
1 |X      X

      123456789
=====
```



## 9. Lucy

**Program Name:** Lucy.java

**Input File:** lucy.dat

Lucy is an artist. Her preferred medium is ASCII characters. She creates large images using nothing more than the characters she finds on her computer's keyboard. She stores these images in data files and sends them to a company called ***Print This!*** to be printed on canvas. Lucy then frames the prints and puts them on sale in her booth down on Guadalupe street in Austin. A problem has arisen with sending the files to ***Print This!*** Lucy's artwork has become so big that it is taking too much time and using too much bandwidth to send the files to the printer. Because you are one of Lucy's most treasured friends and she knows that you are a skilled programmer she has asked if you can write a program that will compress her files and make them easier to send to the printer.

A quick check of the internet turns up an algorithm that looks like it will do the job. It is called **run-length encoding**. Run-length encoding is a very simple "loss-less" data compression routine in which runs of characters (characters that are repeated in sequence) are stored as a pair of values, the character itself and the number of times it occurs in succession. This algorithm works best on data that contains many such runs, simple graphic images such as icons, line drawings, and animations. This algorithm is just what we need to compress Lucy's art work without any loss of each image. Run-length encoding for the string aaaii000000qwwwww\*\*\*\*\* would result in a3i206q1w5\*10.

**Input:** The first line of data will contain a value D representing the number of drawings to compress, followed by D drawing data set. Each drawing data set consists of a number N on one line representing the number of lines in the drawing, followed by N lines that make up the drawing. Each line will contain non-digit alpha and symbol characters, including spaces..

**Output:** For each of the drawings, N lines that consist of the run-length encoding for that line in the original drawing. Each encoded drawing should be ended by this line, "=====".

### Sample input:

```
2
13
.....
@.....@
.@.....@
..@.....@..
...@@@@@@@@@@@@@...
....@.....@....
.....**.....@
.....**.....@
.....@.....@
..../@.....++.....@
.../ &&@.....++.....@
.. / &&& / @ @ @ @ @
./ &&& / .....
8
.....
.....
.....|.....|.....
.....|.....\.....\.....\.....\.....
/\.....|.....|.....\.....\.....\.....
\.....(.....\.....\.....(.....\.....
.....\.....\.....
.....
```

(Lucy – cont)

Sample output:

```
.20
@1.18@1
.1@1.16@1.1
.2@1.14@1.2
.3@14.3
.3@1.12@1.3
.4@1.2*2.2*2.2@1.4
.4@1.2*2.2*2.2@1.4
.5@1.8@1.5
.4/1@1.3+2.3@1.5
.3/1&2@1.2+2.2@1.6
.2/1&3/1@6.7
.1/1&3/1.14
=====
.30
.4_4.1_1.20
.4|1_4|1_4_1_3_2_7.3
.4|1_4|1_2_2\2_2\1/1_1/1\1_2_2\1.2
/1\1_2|1_4|1/1_1_2_1\2_3/1_2/1_1_2_1\1_1
\1_8(1_4_2/1\1_1/1_2(1_4_2/1
.14\1/1.11\1/1.1
.30
=====
```

## 10. Micaela

**Program Name:** Micaela.java

**Input File:** micaela.dat

Micaela is fascinated with writing encrypted messages and has come up with one she thinks will do the trick. See if you can figure out a program to decipher her message.

For example, she wants to send the word “BLUE” as a part of a message to her friend, Ayden, and decides to use two long nonsense words, both of which contain the word “BLUE”, with those four letters in that order, somewhere inside each word, but with many other letters in between. She designs each word so that the word “BLUE” is the longest possible sequence of common letters among both words.

The two code words she sends are **DEBQGLBFUE** and **BMANOPLRSTUEVE**. There are other sequences that are common, but are not as long. For example, **BUE** and **LUE** are both common, but aren’t the longest ones. She is also careful to make sure that there are no two “different” common words that are of the longest length, for obvious reasons; it would be quite confusing to the one receiving the coded message!

To mark the end of a thought in the message, she includes two encoded sequences that have nothing in common. In that case, for testing purposes, she just says “NONE” when that happens.

**Input:** A data file of several pairs of encoded character sequences, in all caps, with each pair on a separate line.

**Output:** The word that is the longest sequence of characters common to both encoded sequences, or the word “NONE” if there is no matching sequence.

**Sample Input:**

```
DEBQGLBFUE BMANOPLRSTUEVE
URPWOEGJSDLFJA RARKQENMED
QGWFEDRGQWFD ULIKJOULIKJYUKLIJ
```

**Sample Output:**

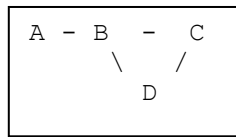
```
BLUE
RED
NONE
```

## 11. Nikita

**Program Name: Nikita.java**

**Input File: nikita.dat**

Nikita's friend Stelios has been researching efficient ways to represent connections within graphs, and has developed a shortest path matrix that shows this. The resulting chart contains the least number of steps it takes to get from one node to any other node. Nikita wants to take Stelios' work and do some analysis about the general nature of any particular graph, especially regarding the accessibility of any node to all other nodes within the graph. For example, he is interested in how many nodes directly connect with each node in the graph, what he considers the "degree" of a node, and generally how "close" each node is to the rest of the graph.



	A	B	C	D
A	0	1	2	2
B	1	0	1	1
C	2	1	0	1
D	2	1	1	0

	1	2
A	1	2
B	3	0
C	2	1
D	2	1

Nikita looks at one of the graphs Stelios has worked with, specifically the four-node graph shown above. The connection matrix in the center shows values that represent how many steps it takes to get to each node from any other node. He observes that the longest path in this situation is 2, and so develops a two-column "length tally" matrix to represent this situation, like the one on the right. This shows that the A node has **one** 1-link connection, and **two** 2-link connections. The B node has **three** 1-link connections, and C and D each have **two** 1-link and **one** 2-link connections.

Just by looking at the graph, Nikita can intuitively judge that the A node seems to be the most isolated, and that B is clearly the most centralized. He thinks about how to mathematically determine this, and comes up with two ways to measure this. First, he wants to measure the "degree" of each node. The formula he comes up with is this:

- Divide the number of 1-link connections for each node by one less than the number of nodes.

Using the length tally matrix shown above, the A node calculation would be  $1/3$ , or 0.33. The B node would be  $3/3$ , or 1.00. The C and D nodes both would be  $2/3$ , or 0.67. Clearly the node with the least degree is A, and the one with the most degree was B, which supports the Nikita's original intuitive observation.

The "closeness" idea is a little more of a challenge, but after pondering a while he decides to use this formula. For each node:

- Find the sum of the number of 1-link connections times 1, 2-link connections times 2, 3-links \* 3, etc.
- Divide that sum **into** the value representing one less than the total number of nodes in the graph.

For node A, the "closeness" calculation would be **3 divided by (1\*1 + 2\*2), or 3/5, or 0.60.**

For node B it would be **3 / (3\*1 + 2\*0), or 3/3, which is 1.00.**

Nodes C and D both have a "closeness" result of **0.67.**

This "closeness" measure at first seems to show the same result as the "degree", in that the A node appears to be the most remote node, with the least "closeness" value, and that B is the most central node, with the greatest "closeness" value, which leaves Nikita unsatisfied. He decides to look at one more example (see next page).

**Nikita - cont**

A - B - C - D - E - F - G - H

He tries these measures on one more graph, essentially a single linked chain of nodes, as shown in the box above. The connection matrix is shown in the center, and the length tally matrix on the right. After analyzing that graph, Nikita finally finds a difference in the “degree” and “closeness” measure, and decides to use both to analyze many more graphs.

	A	B	C	D	E	F	G	H
A	0	1	2	3	4	5	6	7
B	1	0	1	2	3	4	5	6
C	2	1	0	1	2	3	4	5
D	3	2	1	0	1	2	3	4
E	4	3	2	1	0	1	2	3
F	5	4	3	2	1	0	1	2
G	6	5	4	3	2	1	0	1
H	7	6	5	4	3	2	1	0

	1	2	3	4	5	6	7
A	1	1	1	1	1	1	1
B	2	1	1	1	1	1	0
C	2	2	1	1	1	0	0
D	2	2	2	1	0	0	0
E	2	2	2	1	0	0	0
F	2	2	1	1	1	0	0
G	2	1	1	1	1	1	0
H	1	1	1	1	1	1	1

In this example, there is a six way tie (BCDEFG) for the nodes with the greatest “degree” (0.29) of connection, but only two nodes (D and E) share the greatest “closeness” (0.44) measure. The two outermost nodes, A and H, both measure the least in “degree” (0.14) and “closeness” (0.25).

**Input:** Several data sets, each consisting of an initial value N, representing the number of nodes in the graph, followed on the next N lines by an NxN matrix of integer values, representing the connection matrix for the graph as described above, with single space separation.

**Assumptions:** The node labels for each graph will always begin with upper-case A and proceed in alpha order for as many nodes as are represented by the graph.

**Output:** For each data set, output four values, labeled, formatted and aligned exactly as shown below. List all nodes in alpha order that match each measure. Each output will be followed by a single line of three dashes.

**Sample Input:**

```
4
0 1 2 2
1 0 1 1
2 1 0 1
2 1 1 0
8
0 1 2 3 4 5 6 7
1 0 1 2 3 4 5 6
2 1 0 1 2 3 4 5
3 2 1 0 1 2 3 4
4 3 2 1 0 1 2 3
5 4 3 2 1 0 1 2
6 5 4 3 2 1 0 1
7 6 5 4 3 2 1 0
6
0 1 1 2 2 2
1 0 2 1 1 1
1 2 0 3 3 1
2 1 3 0 1 2
2 1 3 1 0 2
2 1 1 2 2 0
```

**Sample Output:**

```
least degree      0.33 A
greatest degree   1.00 B
least closeness    0.60 A
greatest closeness 1.00 B
---
least degree      0.14 AH
greatest degree   0.29 BCDEFG
least closeness    0.25 AH
greatest closeness 0.44 DE
---
least degree      0.40 ACDEF
greatest degree   0.80 B
least closeness    0.50 C
greatest closeness 0.83 B
---
```

## 12. Peter

**Program Name: Peter.java**

**Input File: peter.dat**

Peter is all about efficiency in everything he does. He despises wastefulness, and is always looking for ways to utilize his resources in the most frugal way. This includes managing his money when he spends it, specifically in paying for things he buys with the coins he saves up in a jar on his dresser. He is obsessed with using the fewest number of pieces of money to find exact change for the cost of the item.

He also has a hobby of studying obsolete money systems, or even makes up new ones in his mind, just for fun, still finding ways to make exact change in the most efficient way.

In the American system of coins, using pennies, nickels, dimes, quarters, half-dollars and dollar coins, he would pay for an item costing 70 cents with 1 a half-dollar and two dimes, a total of three coins used.

In an imaginary system, with only three coins valued at 1, 3 and 5 units, an item costing 11 units would use two 5 coins and a 1 coin.

He needs you to write a program to help in testing his efficient change-making efforts.

**Input:** Several sets of data, each set on two lines. The first line contains a single integer C representing the cost of the item to be purchased. The next line contains several integers representing a coin system, always starting with 1, and proceeding upwards in value, with single space separation.

**Output:** The original value C, followed by a value N representing the least number of coins required to make exact change, followed by an ascending list of each coin, all values with single space separation.

**Sample input:**

```
11
1 3 5
70
1 5 10 25 50 100
131
1 4 9 14 22 53
```

**Sample output:**

```
11 3 1 5 5
70 3 10 10 50
131 6 1 1 1 22 53 53
```



# UIL Computer Science Competition

## Invitational B 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**

Number	Name
Problem 1	Baldo
Problem 2	Barb
Problem 3	Constanza
Problem 4	Dilbert
Problem 5	Emily
Problem 6	Fa
Problem 7	Judith
Problem 8	Luann
Problem 9	Lucy
Problem 10	Micaela
Problem 11	Nikita
Problem 12	Peter

**Problem #1**  
**60 Points**

**1. Baldo**

**Program Name: Baldo.java**

**Input File: None**

**Test Output To Screen**

```
A 65 a 97
B 66 b 98
C 67 c 99
D 68 d 100
E 69 e 101
F 70 f 102
G 71 g 103
H 72 h 104
I 73 i 105
J 74 j 106
K 75 k 107
L 76 l 108
M 77 m 109
N 78 n 110
O 79 o 111
P 80 p 112
Q 81 q 113
R 82 r 114
S 83 s 115
T 84 t 116
U 85 u 117
V 86 v 118
W 87 w 119
X 88 x 120
Y 89 y 121
Z 90 z 122
```



**Problem #2**  
**60 Points**

**2. Barb**

**Program Name: Barb.java**

**Input File: barb.dat**

**Test Input File:**

```
red
yellow
green
orange
blue
indigo
violet
```

**Test Output To Screen**

```
red DER er ED
yellow WOLLEY ley LOW
green NEERG erg EEN
orange EGNARO aro NGE
blue EULB lb UE
indigo OGIDNI dni IGO
violet TELOIV oiv LET
```

**Problem #3**  
**60 Points**

### 3. Constanza

**Program Name: Constanza.java**

**Input File: constanza.dat**

**Test Input File:**

yellow	blue	indigo
orange	indigo	green
green	green	red
indigo	red	violet
green	violet	yellow
blue	yellow	indigo
orange	indigo	red
green	red	red
blue	red	yellow
yellow	yellow	orange
red	orange	green
blue	green	indigo
blue	indigo	green
indigo	green	blue
green	blue	yellow
red	orange	orange
violet	green	green
yellow	blue	indigo
indigo	yellow	green
red	red	blue
red	blue	yellow
yellow	blue	orange
orange	indigo	green
green	green	indigo
indigo	red	green
green	violet	blue
blue	yellow	yellow
orange	indigo	orange
green	red	green
blue	red	indigo
yellow	yellow	green
red	red	blue
blue	blue	
	blue	

**Test Output To Screen**

```

red *****
orange *****
yellow *****
green *****
blue *****
indigo *****
violet ****
  
```

**Problem #4**  
**60 Points**

**4. Dilbert**

**Program Name: Dilbert.java**

**Input File: dilbert.dat**

**Test Input File:**

```
Cory Randolph 10/6/2017 25.25
Idalia Saucedo 10/3/2017 15.00
Alonzo Carrasco 10/6/2017 31.10
Steve Flanary 10/2/2017 20.00
Cory Randolph 10/30/2017 18.50
Steve Flanary 10/15/2017 12.75
Cory Randolph 10/14/2017 35.90
Idalia Saucedo 10/20/2017 26.35
Toni Andrews 10/12/2017 19.14
Jeremy Riley 10/4/2017 31.95
Patricia Garza 10/28/2017 52.05
Idalia Saucedo 10/12/2017 15.00
Alonzo Carrasco 10/30/2017 37.75
Steve Flanary 10/16/2017 8.50
Ronald Meads 10/3/2017 18.45
Ronald Meads 10/28/2017 22.22
Elizabeth Perez 10/30/2017 118.90
Bernice Urteaga 10/15/2017 14.99
Patricia Garza 10/20/2017 25.15
Toni Andrews 10/30/2017 85.49
Idalia Saucedo 10/29/2017 9.50
Jeremy Riley 10/27/2017 75.50
```

**Test Output To Screen**

```
Elizabeth Perez
10/30/2017 $118.90
Total Sales $118.90
```

```
Jeremy Riley
10/4/2017 $31.95
10/27/2017 $75.50
Total Sales $107.45
```

```
Toni Andrews
10/12/2017 $19.14
10/30/2017 $85.49
Total Sales $104.63
```

```
Cory Randolph
10/6/2017 $25.25
10/14/2017 $35.90
10/30/2017 $18.50
Total Sales $79.65
```

```
Patricia Garza
10/20/2017 $25.15
10/28/2017 $52.05
Total Sales $77.20
```

**UIL – Computer Science Programming Judge Packet – Invitational B - 2018**

Alonzo Carrasco  
10/6/2017 \$31.10  
10/30/2017 \$37.75  
Total Sales \$68.85

Idalia Saucedo  
10/3/2017 \$15.00  
10/12/2017 \$15.00  
10/20/2017 \$26.35  
10/29/2017 \$9.50  
Total Sales \$65.85

Steve Flanary  
10/2/2017 \$20.00  
10/15/2017 \$12.75  
10/16/2017 \$8.50  
Total Sales \$41.25

Ronald Meads  
10/3/2017 \$18.45  
10/28/2017 \$22.22  
Total Sales \$40.67

Bernice Urteaga  
10/15/2017 \$14.99  
Total Sales \$14.99

Grand Total \$719.44

**Problem #5**  
**60 Points**

**5. Emily**

**Program Name: Emily.java**

**Input File: emily.dat**

**Test Input File:**

```
5 7 3
8 3 12
3 4 5
12 13 5
1 2 3
22 64 89
100 50 25
8 2 1
```

**Test Output To Screen**

```
18
66
17.5
62.5
4.5
3827
1875
5
```

**Problem #6**  
**60 Points**

**6. Fa**

**Program Name: Fa.java**

**Input File: fa.dat**

**Test Input File:**

```
5
1 2 3 2 1
6
3 6 3 3 5 8
6
3 3 3 3 4 3
7
1 2 3 3 2 1 5
11
4 2 3 6 5 8 7 1 4 5 4
10
1 1 1 1 1 1 1 1 1 8
12
7 3 2 3 6 5 4 7 1 4 5 4
10
4 2 3 6 5 8 7 1 4 5
6
1 2 3 4 5 1
5
1 3 5 2 2
21
4 42 27 16 28 3 4 5 9 3 31 5 5 29 10 18 35 35 33 19 41
22
23 8 32 9 5 8 18 35 13 6 7 6 1 11 13 37 2 25 7 28 43 1
2
1 2
3
3 2 1
3
5 5 9
```

**Test Output To Screen**

```
3 ^2 3
12 ^3 13
9 2^3 10
9 3^4 8
20 ^5 21
8 ^8 8
26 5^6 25
20 ^5 17
6 ^3 6
4 ^2 4
182 ^13 191
170 11^12 168
1 0^1 2
3 0^1 3
10 1^2 9
```

**Problem #7**  
**60 Points**

**7. Judith**

**Program Name: Judith.java**

**Input File: judith.dat**

**Test Input File:**

```
395.96774839412213864694087795
375.286439548498
8.412190040221
0.382560882678260591251068
0.23943358855017051969126499
95.96038946102548087733835
0.392863116206040
557.91740143772795279
46.8020528027329
0.91986136436504
0.0000
0.9521580604
773.64490843040425603301021579697
0.187708001820595693
739.1741360315400710907
4.02522642065912585227129617
232.802415492700367772861
0.68050252998169929474
0.3564441495345752345209227900483
807.6684236589081845207172078
197.761868293472
0.3108935600
483.4083054655195578740049479321
50.95293053730
0.468832204284053788334730217824722
0.6056155590844911143307731872
8.3210068983504233163368287
502.4490287116280075170637062189874444
5.8738786083606191739579735324
332.9057251919956969077
0.42649419841162344812229157682254783
```

**Test Output To Screen**

```
0.0000
0.187708001820595693
0.23943358855017051969126499
0.3108935600
0.3564441495345752345209227900483
0.382560882678260591251068
0.392863116206040
0.42649419841162344812229157682254783
0.468832204284053788334730217824722
0.6056155590844911143307731872
0.68050252998169929474
0.91986136436504
0.9521580604
4.02522642065912585227129617
5.8738786083606191739579735324
8.3210068983504233163368287
```

**UIL – Computer Science Programming Judge Packet – Invitational B - 2018**

8.412190040221  
46.8020528027329  
50.95293053730  
95.96038946102548087733835  
197.761868293472  
232.802415492700367772861  
332.9057251919956969077  
375.286439548498  
395.96774839412213864694087795  
483.4083054655195578740049479321  
502.4490287116280075170637062189874444  
557.91740143772795279  
739.1741360315400710907  
773.64490843040425603301021579697  
807.6684236589081845207172078



**Problem #8**  
**60 Points**

**8. Luann**

**Program Name: Luann.java**

**Input File: luann.dat**

**Test Input File:**

```

5
7
1 1
2 2
3 3
4 4
5 5
6 3
7 6
9
1 1
2 4
3 9
4 9
5 9
6 7
7 5

```

**Test Output To Screen**

```

6|      X
5|      X
4|      X
3|  X  X
2|  X
1|X

  1234567
=====
9|  XXX
8|
7|      X
6|
5|      X
4|  X
3|
2|      X
1|X      X

  123456789
=====

```

```

8 2
9 1
4
1 1
2 1
3 1
4 1
1
1 5
6
1 8
2 5
3 6
4 4
5 7
6 3

```

```

1|XXXXX
  1234
=====
5|X
4|
3|
2|
1|

  1
=====
8|X
7|  X
6|  X
5|  X
4|  X
3|    X
2|
1|

  123456
=====

```



Test Output To Screen

```
.20
@1.18@1
.1@1.16@1.1
.2@1.14@1.2
.3@14.3
.3@1.12@1.3
.4@1.2*2.2*2.2@1.4
.4@1.2*2.2*2.2@1.4
.5@1.8@1.5
.4/1@1.3+2.3@1.5
.3/1&2@1.2+2.2@1.6
.2/1&3/1@6.7
.1/1&3/1.14
=====
.30
.4_4.1 1.20
.4|1 4|1_4 1_3 2_7.3
.4|1 4|1_2 2_2 2_1/1 1/1\1_2 2\1.2
/1\1 2|1_4|1/1 1_2 1\2 3/1 2/1 1_2 1\1_1
\1_8(1_4 2/1\1 1/1 2(1_4 2/1
.14\1/1.11\1/1.1
.30
=====
@15
@5_5@5
@4/1>5\1@4
@3/1 1(1.1)1(1.1)1\1@3
@2|1 9|1@2
@3\1 302 2/1@3
@4\1-5/1@4
@15
=====
;11:6;3:5c1o102k1o1c1:10;8
;2011x1k1d2k1@1N4K1d1:1;4:2;10
;1,3;2,4;2,2;611k104@1K1X1N1W2N1O1c1;14,1
,17;3:1o1O1@1O4@1K1X1N2X1x1:1;5,1;5,3
,2'2,9'3,2;3:1x1@4O3@1K1X1N1@111;6,8
,4'1,7'5,2;3:1x1@4O1k1O1@1K1X2O1c1,2;3,2'4,2
'1,12'4,4;2:1d1k1O1@1O2k1O1@1K1X1N1@1d1o2c1:1;1,1'4,2
'4,2'2,13;3:1c11d1x1O2@1K2X1N1W3N1X1@1k1o1;1'2,1;2
,1'8,2;3,4'3,2;2:1;1,1;1o1@1K5X2N2X1K1X1N1K1x1c1;2:1
,1'7,4;2,5'2,3;3,1;1o1K1N1X1K6k1c1;1o1@1X2@1d1:1;1
'8,1'4,3'1,8;2,1;1x1N1W1N6X2O1k1O1@1K3O1o1:1
'6,1;1,1'5,2;1,2'1,5;2,1:1x1N1W2N6X3K1@1O2k1x111;1
,1'1.2'1,3'3,2'1,2;1,2'1,2;5:111@1W3N3X2K2@1O1k1x2d1o111:1,1
;1'1.3'3,1'5,2:111d1x1k1@2K1X2K2X1W5N2X1K1@1O1x1d111c3:2;1,1
'2.3'1,4'2,1c1x1O1X1N1W15N3X2K1@1O1k1d1c1:2;2,2
,2'1.2'1,2'1,1c1x1K1N1W1M1W14N2X6K2@1O1k1o1;1,1;4
'1,2'3,1:1o1O1N1W1M2W11N4X3K6@3O1k1x1:1,1;1,3
'5;1d1X1W13N6X2K8@3O1k2x1:1'1,3'1
'2.1,1o1@1W12N4X4K6@5O3k1x2c1'1.2'2.1
'2,1d1N1W3N1W8N3X2K3@6O5k4d2o1;1.3'2.1
,2c1@1W4N1X2N1W2N3X1N1X3K2@1O2k1x7d1o1d2o3c1'1.3'1.2
c1x1@1N1W1N4X1K2X1N2X1K2X1K1@1O3d111o111;2:2c111c1;111o111c2o2,1.6
;1d1K1N1K1k1x1O1K3@2X1N1X1@2O1d1o1x2d1c1,1;1'1.3'1;1:1'2c112:1'1,1:2'1.2'1.2
.1,111k1x111c2d1O1@2O2@1k1d2o1:1c1o211;1'1.4'1,2.2'1;1,1;2.2'1:1;1.4,1
.3;2,1'2,1;111o113:2c1:1,1:111o1c1,1'2.2'1,2.4'1.1,2.3'1;1'4,1
.9'1,3;1'3,1.1'1;1:2,1.4'3.11'2.3'1,1'1
1.8,2.4'2.3'3.19'2.1'2,2.1
=====
```

**Problem #10**  
**60 Points**

**10. Micaela**

**Program Name: Micaela.java**

**Input File: micaela.dat**

**Test Input File:**

```
DEBQGLBFUE BMANOPLRSTUEVE
URPWOEGJSDLFJA RARKQENMED
QGWFEDRGQWFD ULIKJOULIKJYUKLIJ
QESSJYUBNGVLHXSLDCBWXBOMUOWWHJZNUUEUAF HSKCRTYJJL
TIMSMXWBEYJZQFWUWPGRQIEIXQ CNUPMBJADQQCVBJOHIFMTX
LKATSHXQATSRTJFLKHUAQLXD LIKKUINCMG
RVLOXRHSBFNOZESJTKAFY TWTPMFNUDQREEOIHBQDUGPHIUTCAOIZSDDCHUMBV
NOPQRSTUVWXYZRSTUVWX ABCDEFGHIJKLMLKJABCEFDILM
GMALAKZQYAJPYAOXXQFPUOEFLVXOACZMZBRSKCT
BQAIFIKPMHSZTQMHVKNZDQKHKKZCTKZUHWNCQVRGAVPMY
EJTZWCBMVBJIINHYVFDSUSVWUCWXBVJJDXWZRHLFUWISBFVK FOXBIETRCPJMPATYHQRBIIPAYCEB
CFJTPWGCQXZNVBMTKXKDYVUZNONSTONYSUFDT AKAJVIPLVPFUTVZHCW
KINHFWITRNQMHQCIPIHIBSTKLEWRWMP CTSYGSZQJEHTBDXOKBXOGCWUPBEMBFUX
MVIJZTQXTCUNQIIJSAGDGZRP GICCMFVQPIGKJ
GSRRAJNXUKYHPKFOBFNNPBKFLFBEPKJXBKVYJLOZNNUYVONNU
ECQSBMZPAYBLPJOOHVDSKOIVTATE NVNIBNJOEPNHRKGQHZZWWJ
```

**Test Output To Screen**

```
BLUE
RED
NONE
HSCJ
MBJQQIX
LKKU
ROHBOZS
NONE
QAFPMZMKCT
ETCMBIIYCB
JPVTVZ
TQHCPBEM
JXUNPFPKJ
BJOHK
```

**Problem #11**  
**60 Points**

## 11. Nikita

Program Name: Nikita.java

Input File: nikita.dat

Test Input File:

```

4
0 1 2 2
1 0 1 1
2 1 0 1
2 1 1 0
8
0 1 2 3 4 5 6 7
1 0 1 2 3 4 5 6
2 1 0 1 2 3 4 5
3 2 1 0 1 2 3 4
4 3 2 1 0 1 2 3
5 4 3 2 1 0 1 2
6 5 4 3 2 1 0 1
7 6 5 4 3 2 1 0
6
0 1 1 2 2 2
1 0 2 1 1 1
1 2 0 3 3 1
2 1 3 0 1 2
2 1 3 1 0 2
2 1 1 2 2 0
6
0 1 3 2 3 3
1 0 2 1 2 2
3 2 0 1 2 2
2 1 1 0 1 1
3 2 2 1 0 1
3 2 2 1 1 0
5
0 1 2 1 2
1 0 2 1 2
2 2 0 1 2
1 1 1 0 1
2 2 2 1 0
7
0 1 2 2 3 2 1
1 0 1 1 2 2 2
2 1 0 1 2 1 2
2 1 1 0 1 2 3
3 2 2 1 0 1 2
2 2 1 2 1 0 1
1 2 2 3 2 1 0
9
0 1 2 3 0 0 0 0 0
1 0 1 2 0 0 0 0 0
2 1 0 1 0 0 0 0 0
3 2 1 0 0 0 0 0 0
0 0 0 0 0 1 0 0 1
0 0 0 0 1 0 0 0 2
    
```

# UIL – Computer Science Programming Judge Packet – Invitational B - 2018

```

0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 1 0 0
0 0 0 0 1 2 0 0 0
12
0 1 2 2 4 5 2 3 3 2 1 1
1 0 1 1 3 4 2 3 2 1 1 2
2 1 0 1 4 5 3 4 3 2 2 3
2 1 1 0 4 5 3 4 3 2 2 3
4 3 4 4 0 1 4 5 1 2 3 5
5 4 5 5 1 0 5 6 2 3 4 6
2 2 3 3 4 5 0 1 3 2 1 3
3 3 4 4 5 6 1 0 4 3 2 4
3 2 3 3 1 2 3 4 0 1 2 4
2 1 2 2 2 3 2 3 1 0 1 3
1 1 2 2 3 4 1 2 2 1 0 2
1 2 3 3 5 6 3 4 4 3 2 0
25
0 1 2 2 4 5 2 3 3 2 1 1 1 1 2 3 3 5 6 3 4 4 3 2 2
1 0 1 1 3 4 2 3 2 1 1 2 2 2 1 2 2 4 5 3 4 3 2 2 3
2 1 0 1 4 5 3 4 3 2 2 3 3 3 2 1 2 5 6 4 5 4 3 3 4
2 1 1 0 4 5 3 4 3 2 2 3 3 3 2 2 1 5 6 4 5 4 3 3 4
4 3 4 4 0 1 4 5 1 2 3 5 5 5 4 5 5 1 2 5 6 2 3 4 6
5 4 5 5 1 0 5 6 2 3 4 6 6 6 5 6 6 2 1 6 7 3 4 5 7
2 2 3 3 4 5 0 1 3 2 1 3 3 3 3 4 4 5 6 1 2 4 3 2 4
3 3 4 4 5 6 1 0 4 3 2 4 4 4 4 5 5 6 7 2 1 5 4 3 5
3 2 3 3 1 2 3 4 0 1 2 4 4 4 3 4 4 2 3 4 5 1 2 3 5
2 1 2 2 2 3 2 3 1 0 1 3 3 3 2 3 3 3 4 3 4 2 1 2 4
1 1 2 2 3 4 1 2 2 1 0 2 2 2 2 3 3 4 5 2 3 3 2 1 3
1 2 3 3 5 6 3 4 4 3 2 0 2 2 3 4 4 6 7 4 5 5 4 3 1
1 2 3 3 5 6 3 4 4 3 2 2 0 2 3 4 4 6 7 4 5 5 4 3 3
1 2 3 3 5 6 3 4 4 3 2 2 2 0 3 4 4 6 7 4 5 5 4 3 3
2 1 2 2 4 5 3 4 3 2 2 3 3 3 0 3 3 5 6 4 5 4 3 3 4
3 2 1 2 5 6 4 5 4 3 3 4 4 4 3 0 3 6 7 5 6 5 4 4 5
3 2 2 1 5 6 4 5 4 3 3 4 4 4 3 3 0 6 7 5 6 5 4 4 5
5 4 5 5 1 2 5 6 2 3 4 6 6 6 5 6 6 0 3 6 7 3 4 5 7
6 5 6 6 2 1 6 7 3 4 5 7 7 7 6 7 7 3 0 7 8 4 5 6 8
3 3 4 4 5 6 1 2 4 3 2 4 4 4 4 5 5 6 7 0 3 5 4 3 5
4 4 5 5 6 7 2 1 5 4 3 5 5 5 5 6 6 7 8 3 0 6 5 4 6
4 3 4 4 2 3 4 5 1 2 3 5 5 5 4 5 5 3 4 5 6 0 3 4 6
3 2 3 3 3 4 3 4 2 1 2 4 4 4 3 4 4 4 5 4 5 3 0 3 5
2 2 3 3 4 5 2 3 3 2 1 3 3 3 3 4 4 5 6 3 4 4 3 0 4
2 3 4 4 6 7 4 5 5 4 3 1 3 3 4 5 5 7 8 5 6 6 5 4 0

```

## Test Output To Screen

```

least degree      0.33 A
greatest degree   1.00 B
least closeness    0.60 A
greatest closeness 1.00 B
---
least degree      0.14 AH
greatest degree   0.29 BCDEFG
least closeness    0.25 AH
greatest closeness 0.44 DE
---
least degree      0.40 ACDEF
greatest degree   0.80 B

```

**UIL – Computer Science Programming Judge Packet – Invitational B - 2018**

```
least closeness      0.50 C
greatest closeness  0.83 B
---
least degree         0.20 AC
greatest degree     0.80 D
least closeness      0.42 A
greatest closeness  0.83 D
---
least degree         0.25 CE
greatest degree     1.00 D
least closeness      0.57 CE
greatest closeness  1.00 D
---
least degree         0.33 AEG
greatest degree     0.50 BCDF
least closeness      0.55 AEG
greatest closeness  0.67 BCF
---
least degree         0.13 ADFGHI
greatest degree     0.25 BCE
least closeness      1.33 AD
greatest closeness  8.00 GH
---
least degree         0.09 FHL
greatest degree     0.45 B
least closeness      0.24 F
greatest closeness  0.52 BK
---
least degree         0.04 MNOPQRSTUVWXYZ
greatest degree     0.25 B
least closeness      0.18 S
greatest closeness  0.43 BK
---
```

**Problem #12**  
**60 Points**

## 12. Peter

**Program Name: Peter.java**

**Input File: peter.dat**

**Test Input File:**

```
11
1 3 5
70
1 5 10 25 50 100
131
1 4 9 14 22 53
5
1 3 5
8
1 3 5
399
1 5 10 25 50 100
231
1 4 9 14 22 53
98
1 4 9 14 22 53
99
1 4 9 14 22 53
100
1 4 9 14 22 53
101
1 4 9 14 22 53
103
1 4 9 14 22 53
106
1 4 9 14 22 53
107
1 4 9 14 22 53
99999
1 99990
```

**Test Output To Screen**

```
11 3 1 5 5
70 3 10 10 50
131 6 1 1 1 22 53 53
5 1 5
8 2 3 5
399 11 1 1 1 1 10 10 25 50 100 100 100
231 7 1 4 14 53 53 53 53
98 4 1 22 22 53
99 5 1 1 22 22 53
100 6 1 1 1 22 22 53
101 4 4 22 22 53
103 6 1 1 4 22 22 53
106 2 53 53
107 3 1 53 53
99999 10 1 1 1 1 1 1 1 1 1 99990
```