

UIL COMPUTER SCIENCE WRITTEN TEST

2018 REGION

APRIL 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 REGION

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 is not equal to $AF_{16} + 77_8$?

- A) 22201_3 B) EE_{16} C) 238_{10} D) 356_8 E) 11101110_2

Question 2.

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

- A) -1 B) 1 C) 5 D) -5 E) 0

```
out.println(-8%3+15%-4);
```

Question 3.

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

- A) %, -12345.33
B) (12, 345.33)
C) -12345.3
D) -12345.33
E) -12, 345.33

```
out.printf("%,7.2f",-12345.329);
```

Question 4.

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

- A) 18 B) 19 C) 20
D) 21 E) -1

```
String s="thirty seconds to mars";
int x=s.indexOf(s.substring(18, 19),20);
out.print(x);
```

Question 5.

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

- A) true B) false

```
boolean b=false;
out.print(b||b&&b^b);
```

Question 6.

Which of the following may replace **<code>** in the line of code shown on the right to ensure that it will compile?

- A) int B) float C) double D) long
E) More than one of the above.

```
<code> x=Math.round(Math.PI);
```

Question 7.

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

- A) m B) n C) o D) p E) q

```
double d='c'+12.75;
out.print((char)d);
```

Question 8.

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

- A) 32
B) 43
C) 33
D) 38
E) 37

```
int n=1593535253;
int s=0;
while(n>0) {
    switch(n%10) {
        case 3:s++;break;
        case 5:s+=n%10;break;
        default:s+=5;
    }
    n/=10;
}
out.print(s);
```

<div>Question 9.</div> <div>What is the output of the code segment shown on the right?</div> <div><div>A) 1 2 3 4</div><div>B) 0 1 2 3 4</div><div>C) 1 2 3 4 5</div><div>D) 0 1 2 3 4 5</div><div>E) 0 2 4</div></div>	<pre>int k=0; while(++k<5) out.print(k+" ");</pre>	
<div>Question 10.</div> <div>What is the output of the code segment to the right?</div> <div><div>A) [6, 7, 8, 12, 10]</div><div>[1, 11, 3, 4, 5]</div><div>B) [1, 11, 3, 12, 5]</div><div>[1, 11, 3, 12, 5]</div><div>C) [6, 11, 8, 12, 10]</div><div>[6, 11, 8, 12, 10]</div><div>D) [1, 11, 3, 4, 5]</div><div>[6, 7, 8, 12, 10]</div><div>E) Error. Throws ArrayIndexOutOfBoundsException.</div></div>	<pre>int a[]= {1,2,3,4,5}; int b[]= {6,7,8,9,10}; b=a; a=b; b[1]=11;a[3]=12; out.println(Arrays.toString(a)); out.println(Arrays.toString(b));</pre>	
<div>Question 11.</div> <div>Which of the following may not replace <code><code></code> in the code segment shown on the right?</div> <div><div>A) <code>nextInt()</code></div><div>B) <code>nextDouble()</code></div><div>C) <code>next()</code></div><div>D) <code>nextFloat()</code></div><div>E) None of the above. All may replace <code><code></code>.</div></div>	<pre>Scanner s=new Scanner("2 12 6 8 1"); double sum=0; while(s.hasNext()) sum+=s.<code>; out.print(sum);</pre>	
<div>Question 12.</div> <div>Which of the following code segments will find and print the sum of the odd numbers within the array <code>a</code>? Array <code>a</code> has been initialized with this line : <code>int[] a= {5,6,7,4,1,9,8};</code></div>		
<div>A.</div> <pre>int sum=0; int x=0,y=a.length; while(x<=y) { if(a[x]%2!=0) sum+=a[x]; x++; } out.print(sum);</pre>	<div>B.</div> <pre>int sum=0; for(int x=0;x<a.length;x++) if(a[x]%2==0) sum+=a[x]; out.print(sum);</pre>	<div>C.</div> <pre>int sum=0; for(int x=1;x<=a.length;x++) if(a[x]%2!=0) sum+=a[x]; out.print(sum);</pre>
<div>D.</div> <pre>int sum=0; int x=0,y=a.length; while(x<y) if(a[x]/2!=0) sum+=a[x]; x++; out.print(sum);</pre>	<div>E.</div> <pre>int sum=0; for(int x=0;x<a.length;x++) if(a[x]%2!=0) sum+=a[x]; out.print(sum);</pre>	

<p>Question 13.</p> <p>What is the output of the code segment to the right?</p> <p>A) 7 B) 21 C) 4 D) 0 E) Error. The operator is undefined for the <code>int</code> data type.</p>	<pre>int g=5,h=2,i=3; out.print(g h*i);</pre>
<p>Question 14.</p> <p>What is printed by the segment of code shown on the right?</p> <p>A) 2147450880 B) 2147450881 C) -32895 D) 32895 E) 32640</p>	<pre>int i=Short.MAX_VALUE-Byte.MIN_VALUE; out.print(i);</pre>
<p>Question 15.</p> <p>What is the output of the code segment to the right?</p> <p>A) [5, 6, 4, 8] B) [5, 6] C) [5, 6, 4] D) [5, 6, 8] E) Error. Throws an <code>IndexOutOfBoundsException</code>.</p>	<pre>ArrayList<Integer> a=new ArrayList<Integer>(); a.add(5); a.add(3); a.add(8); a.add(1, 6); a.set(2, 4); a.remove(3); out.print(a);</pre>
<p>Question 16.</p> <p>What is the output of the client code shown to the right?</p> <p>A) gouiluilill B) liluilgouil C) ggogouil D) gouilgog E) ggogougouigouil</p>	<pre>public static String rec(String s) { if(s.length()==1) return s; else return rec(s.substring(s.length()/2))+s; } //client code String s="gouil"; out.print(rec(s));</pre>
<p>Question 17.</p> <p>What is the output of the code segment listed on the right?</p> <p>A) [8, 9, 12] [3, 4, 11] [9, 10, 11] B) [8, 2, 12] [9, 3, 13] [10, 6, 14] C) [8, 9, 10] [3, 4, 5] [9, 10, 11] D) [8, 2, 12] [9, 3, 13] [10, 4, 14] E) There is no output due to an error.</p>	<pre>int[][] mat= {{8,2,12}, {3,1,11}, {9,6,0}}; for(int x=0;x<mat.length;x++) for(int y=1;y<mat[x].length;y++) if(mat[x][y]<=mat[x][y-1]) mat[x][y]=mat[x][y-1]+1; for(int[] m:mat) out.println(Arrays.toString(m));</pre>

<p>Question 18.</p> <p>The line shown on the right is an example of _____.</p> <ul style="list-style-type: none"> A) scope restriction B) parsing C) coupling D) casting E) auto-boxing 	<pre>Integer i=35;</pre>
<p>Question 19.</p> <p>Which of the following must replace <code> in class Cls?</p> <ul style="list-style-type: none"> A) extends B) implements C) inherits D) overrides E) overloads 	<p>//Use the following code to answer questions //19 - 22.</p> <pre>public interface Intf { public String m1(String s); public boolean m2(String s,int i); public int m3(String s,char c); }</pre>
<p>Question 20.</p> <p>Which of the methods shown in class Cls are required?</p> <ul style="list-style-type: none"> A) m1 B) m2 C) m3 D) All of the above. E) None of the above. 	<pre>public class Cls <code> Intf { public String m1(String s) { return null; } public boolean m2(String s, int i) { return false; } public int m3(String s, char c) { return 0; } }</pre>
<p>Question 21.</p> <p>All of the methods shown within the Intf interface are _____?</p> <ul style="list-style-type: none"> A) static B) comparable C) abstract D) extendable E) More than one of the above. 	<pre>//Client code Intf i=new Cls(); Cls c=new Cls(); out.print(i instanceof Cls); out.print(c instanceof Intf);</pre>
<p>Question 22.</p> <p>Assuming that <code> has been correctly replaced, what is the output of the client code shown on the right?</p> <ul style="list-style-type: none"> A) true>true B) true>false C) false>true D) false>false E) There is no output due to an error. 	<pre>//Client code Intf i=new Cls(); Cls c=new Cls(); out.print(i instanceof Cls); out.print(c instanceof Intf);</pre>

Question 23.

What is printed by the code shown on the right?

- A) 01110
- B) 11100
- C) 111
- D) 1
- E) 10000

```
int x=7<<2;
out.print(Integer.toBinaryString(x));
```

Question 24.

The method `sort` shown on the right is intended to implement the Quicksort algorithm. Which of the following should replace **<code 1>** so that the middle element in the partition is assigned to `z`?

- A) `list[(right-left)/2]`
- B) `list[list.length/2]`
- C) `left+right/2`
- D) `list[(left+right)]`
- E) `list[(left+right)/2]`

Question 25.

What is the purpose of **line #1**?

- A) Iterate until a value less than the pivot value is found.
- B) Iterate until a value greater than the pivot value is found.
- C) Change the pivot value to be the element stored in `list[x]`.
- D) Find the smallest value in the left hand partition of `list`.
- E) Find the largest value in the right hand partition of the `list`.

//Use the code for method `sort` to answer questions 24, 25, 26 and 27.

```
public static void sort(int list[],int left,
                        int right)
{
    if(left>=right) return;
    int x=left;
    int y=right;
    int z=<code 1>;
    while(x<y) {
        while(list[x]<z) x++;//line #1
        while(list[y]>z) y--;
        if(x<=y) {
            int t=list[x];
            list[x]=list[y];
            list[y]=t;
            x++;
            y--;
        }
    }
    <code 2>
    <code 3>
}
```

Question 26.

Which of the following lines of code must replace **<code 2>** and **<code 3>** so that `sort` will compile and execute correctly?

- A) `sort(list,y,left);`
`sort(list,x,right);`
- B) `sort(list,left,y);`
`sort(list,right,x);`
- C) `sort(list,left,y);`
`sort(list,x,right);`
- D) `sort(list,left,right);`
`sort(list,x,y);`
- E) `sort(list,left,right);`
`//Only one line of code is needed.`

Question 27.

Once fully and correctly implemented what is the best case relative time complexity (Big-O value) of the method `sort`?

- A) $O(1)$
- B) $O(n)$
- C) $O(n^2)$
- D) $O(n \log n)$
- E) $O(\log n)$

<p>Question 28.</p> <p>Which of the following must replace <code> in the code fragment shown on the right to ensure that the result of the expression will be a random whole number between 20 and 29 inclusive?</p> <p>A) *10+20 B) +10*20 C) *20+10 D) *10+29 E) *29+10</p>	<pre>(int) (Math.random()<code>)</pre>
<p>Question 29.</p> <p>Which of the following can replace <code> in the segment shown on the right?</p> <p>A) <String> B) <Object> C) Will compile and run without replacing <code>. D) A and B E) A, B, and C</p>	
<p>Question 30.</p> <p>What is the output at line #1 in the code segment shown on the right?</p> <p>A) [] B) [turtle, cat, zebra] C) [bird, antelope] D) [bird, turtle, antelope, cat, zebra] E) [dog, cat, zebra, turtle, monkey]</p>	<pre>//Use the following code segment to answer //questions 29, 30 and 31. Queue<code> p=new LinkedList<code>(); p.add("bird");p.add("turtle");p.add("antelope"); p.add("cat");p.add("zebra"); ArrayList<code> list=new ArrayList<code>(); list.add("dog");list.add("cat");list.add("zebra"); list.add("turtle");list.add("monkey"); p.removeAll(list); out.println(p);//line #1 out.println(list);//line #2</pre>
<p>Question 31.</p> <p>What does line #2 in the code segment shown on the right print?</p> <p>A) [dog, cat, zebra, turtle, monkey] B) [dog, monkey] C) [cat, zebra, turtle] D) [cat, dog, monkey, turtle, zebra] E) []</p>	
<p>Question 32.</p> <p>What is the output of the code segment shown on the right?</p> <p>A) 5 8 5 B) 5 8 8 C) 4 8 7 D) 4 7 7 E) 5 8 11</p>	<pre>int c=5,d=7,e=3; if(++d-c<=e c--<d+c) e+=c; else e+=d; out.print(c+" "+d+" "+e);</pre>

Question 33.

What is printed by the code segment shown here?

```
String s1="supper";
String s2="superduper";
String s3=s1.substring(1, 4).equals(s2.substring(1, 4))?s1:s2;
out.print(s3);
```

- A) uppupe
- B) upp
- C) upe
- D) superduper
- E) supper

Question 34.

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

- A) [5, 8, 4, 1, 2, 3, 6, 7, 9]
- B) [1, 2, 3, 4, 5, 6, 7, 8, 9]
- C) [5, 8, 2, 4, 6, 7, 9, 1, 3]
- D) [5, 8, 4, 1, 2, 6, 7, 9, 3]
- E) [5, 8, 4, 2, 6, 7, 9, 1, 3]

```
int[] list= {5,8,4,6,2,9,7,1,3};
Arrays.sort(list, 3, 7);
out.print(Arrays.toString(list));
```

Question 35.

Which of the following best describes the value returned by method?

- A) The number of vowels in the string *s* or -1 if *s* does not contain a vowel.
- B) The index number of the first vowel contained in *s* or -1 if *s* does not contain a vowel.
- C) The index value of each vowel in *s* or -1 if there are no vowels present.
- D) The ordinal (1 based) position of the first vowel contained in *s* or -1 if *s* does not contain a vowel.
- E) The number of times the string "aeiou" is found in *s* or -1 if it does not occur.

```
public static int method(String s) {
    s=s.toLowerCase();
    int x;
    for(x=0;x<s.length();x++)
        if(s.substring(x, x+1).matches("[aeiou]"))
            break;
    if(x!=s.length())
        return x+1;
    else
        return -1;
}
```

Question 36.

Which of the following Boolean expressions is not equivalent to the others listed?

- A) $A * (A + B)$
- B) $A + A * B$
- C) $A + 1$
- D) $A * 1$
- E) $A + 0$

Question 37.

The weight in a weighted graph is a property of the graph's _____?

- A) edges
- B) vertices
- C) direction
- D) cycle
- E) spanning tree

Question 38.

Which of the following is the decimal equivalent of the signed 8-bit binary two's complement value shown on the right?

- A) 121
- B) -6
- C) -134
- D) 134
- E) -122

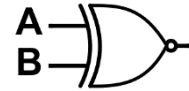
10000110

Question 39.

Rewrite this postfix expression using infix notation. $A B C * + D -$

Question 40.

Write the equivalent Boolean expression for the diagram shown here? You may use Java or generic notation to write the answer.



★ ANSWER KEY – CONFIDENTIAL ★

UIL COMPUTER SCIENCE – 2018 REGION

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

- | | | | |
|------------------|------------------|------------------|--|
| 1) <u> A </u> | 11) <u> C </u> | 21) <u> C </u> | 31) <u> A </u> |
| 2) <u> B </u> | 12) <u> E </u> | 22) <u> A </u> | 32) <u> B </u> |
| 3) <u> E </u> | 13) <u> A </u> | 23) <u> B </u> | 33) <u> D </u> |
| 4) <u> E </u> | 14) <u> D </u> | 24) <u> E </u> | 34) <u> E </u> |
| 5) <u> B </u> | 15) <u> C </u> | 25) <u> B </u> | 35) <u> D </u> |
| 6) <u> E </u> | 16) <u> B </u> | 26) <u> C </u> | 36) <u> C </u> |
| 7) <u> C </u> | 17) <u> A </u> | 27) <u> D </u> | 37) <u> A </u> |
| 8) <u> D </u> | 18) <u> E </u> | 28) <u> A </u> | 38) <u> E </u> |
| 9) <u> A </u> | 19) <u> B </u> | 29) <u> E </u> | *39) <u> A+B*C-D </u> |
| 10) <u> B </u> | 20) <u> D </u> | 30) <u> C </u> | *40) <u> !(A^B) also $\overline{A \oplus B}$ </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.	A	$AF_{16}=175_{10}$ $77_8=63_{10}$ $175+63=238_{10}$ Answer choices B, C, D, and E are all equal to 238_{10} $22201_3 = 235_{10}$ $(1 \times 1) + (2 \times 9) + (2 \times 27) + (2 \times 81) = 235$																																	
2.	B	$-8\%3+15\%-4 =$ $-2+3 =$ 1																																	
3.	E	The escape sequences \" surround the result in quotes. The flag , (comma) makes the number have a comma separator. The flag ((parenthesis) encloses the number in parenthesis if it is negative. 7.2 rounds the number to two decimal places and places it into 7 spaces. When not enough space is allocated, printf will take the space that is needed.																																	
4.	E	The substring method returns m. The two argument indexOf method will return the index of the first argument when it is found after the index number provided by the second argument. In this case, m does not appear in the string after index 20.																																	
5.	B	$F F\&\&F^F=$ $F F\&\&F=$ $F F=$ F																																	
6.	E	Math.round(x) returns long. long is promoted to float and double when needed.																																	
7.	C	ASCII value of 'c' is 99. $99+12.75=111.75$. Casting to char truncates to 111. ASCII number 111 = 'o'.																																	
8.	D	<table><thead><tr><th>n</th><th>n%10</th><th>s</th></tr></thead><tbody><tr><td>1593535253</td><td>3</td><td>1</td></tr><tr><td>159353525</td><td>5</td><td>6</td></tr><tr><td>15935352</td><td>2</td><td>11</td></tr><tr><td>1593535</td><td>5</td><td>16</td></tr><tr><td>159353</td><td>3</td><td>17</td></tr><tr><td>15935</td><td>5</td><td>22</td></tr><tr><td>1593</td><td>3</td><td>23</td></tr><tr><td>159</td><td>9</td><td>28</td></tr><tr><td>15</td><td>5</td><td>33</td></tr><tr><td>1</td><td>1</td><td>38</td></tr></tbody></table>	n	n%10	s	1593535253	3	1	159353525	5	6	15935352	2	11	1593535	5	16	159353	3	17	15935	5	22	1593	3	23	159	9	28	15	5	33	1	1	38
n	n%10	s																																	
1593535253	3	1																																	
159353525	5	6																																	
15935352	2	11																																	
1593535	5	16																																	
159353	3	17																																	
15935	5	22																																	
1593	3	23																																	
159	9	28																																	
15	5	33																																	
1	1	38																																	
9.	A	k is incremented before the comparison to 5 is made, therefore, 0 is not printed.																																	
10.	B	After the line b=a; both a and b point to the same array, array a. From this point any action taken on either array is taken on array a.																																	
11.	C	next() returns a string.																																	
12.	E	A produces an out of bounds error. B finds the sum of the evens. C produces an out of bounds error. D finds the sum of all the numbers except for 1.																																	
13.	A	$5 2*3=5 6.$ <table><tr><td></td><td>101</td></tr><tr><td>OR</td><td>110</td></tr><tr><td>=</td><td>111</td></tr></table> $111_2 = 7$		101	OR	110	=	111																											
	101																																		
OR	110																																		
=	111																																		
14.	D	$32767-(-128)=32895$																																	
15.	C	5 5 3 5 3 8 5 6 3 8 5 6 4 8 5 6 4																																	
16.	B	When the base case is reached the call stack is: l il uil gouil Pop each off the stack to get: liluilgouil																																	
17.	A	For each row, if the current value in the row is less than the previous one, replace the current one with the previous one +1.																																	

18.	E	Auto boxing automatically converts a primitive data type value to an object.
19.	B	Classes implement interfaces.
20.	D	When a class implements an interface, that class must implement all of the methods defined in the interface.
21.	C	An abstract method has a signature but is not implemented.
22.	A	i is a Cls object, c is a Cls object that has inherited from Intf.
23.	B	$7 = 00111_2$ Shift the bits to the left two spaces. That leaves 11100_2 .
24.	E	The partition being sorted is bounded by the index values stored in left and right.
25.	B	This value will be swapped with the value found by the next line which will be smaller than the pivot.
26.	C	Divide this partition in two and recursively call the sort method on each of the new partitions. Since x and y have crossed paths in the code, ending the while loop, y is now the upperbound of the left partition and x is the lowerbound of the right partition.
27.	D	Best $O(n \log n)$, Average $O(n \log n)$, Worst $O(n^2)$
28.	A	Math.random returns 0.0 (inclusive) to 1.0 (exclusive). $0.0 * 10 + 20 = 0 + 20 = 20$ $0.99 * 10 + 20 = 9.9 + 20 = 29.9$ which casts to 29
29.	E	Object is the cosmic super class. The elements in each collection are Strings. While it is greatly preferred that they are, collections do not strictly have to be parameterized.
30.	C	removeAll method removes everything in list from p.
31.	A	list is unaffected by the call to removeAll.
32.	B	The left hand operand of the operator evaluates to true. Because of short circuit evaluation the right hand side is never evaluated. Therefore c is not decremented.
33.	D	s1.substring(1,4) is "upp". s2.substring(1,4) is "upe". Ternary operator works like this: condition?true case:false case. "upp" is not equal to "upe" so false case is s2.
34.	E	This version of sort sorts the array beginning at the first argument and ending at one less than the second argument.
35.	D	[aeoiu] will match any of the vowels in the set. The loop breaks when a vowel is found. x stores the index number of that vowel. x+1 is the actual position of the vowel within the word when starting the count at one.
36.	C	$A * (A + B) = A$ - Law of Absorption $A + A * B = A$ - Law of Absorption $A + 1 = 1$ - Law of Union $A * 1 = A$ - Identity Law for AND $A + 0 = A$ - Identity Law for OR
37.	A	The weight of an edge is the cost to travel along that edge from one vertex to another. The cost can be measured in money, time, distance or any other appropriate unit.
38.	E	Find the complement: 01111001. Add one to get 01111010. Convert to a decimal to get 122. Has to be negative because the sign bit was 1 in the original value. So, $10000110_2 = -122$.
39.		$A + B * C - D$ or $B * C + A - D$ or $A - D + B * C$
40.		This is the logic gate for XNOR. Also accept $\overline{A \oplus B}$. Do not accept XNOR as an answer.

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"/>



Computer Science Competition Region 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	Ariana
Problem 2	Bingwen
Problem 3	Caleb
Problem 4	Debra
Problem 5	Emily
Problem 6	Joe
Problem 7	Opus
Problem 8	Prashant
Problem 9	Ramya
Problem 10	Samantha
Problem 11	Vincent
Problem 12	William

1. Ariana

Program Name: Ariana.java

Input File: ariana.dat

Ariana is an avid Harry Potter fan and recently read about the Black Punishment Quill that Dolores Umbridge invented and inflicted on Harry several times while she served as the Hogwarts High Inquisitor in the book, **Order of the Phoenix**. According to the *Harry Potter Lexicon*, the quill is "**A long, thin black quill with an unusually sharp point, this quill does not use ink. Instead, when used to write, it magically slices into the back of the hand holding it and draws blood in place of ink. The cuts at first heal immediately, but over a long session the damage will be repaired more and more slowly, until the hand is finally left raw and bleeding.**"

The sentence she always instructed Harry to write, over and over again, until his hand was bleeding and eventually scarred permanently, was, "*I must not tell lies.*"

In one gruesome scene of the story which eventually was cut from the book, Professor Umbridge was particularly angry with Harry, and insisted that he not only write the sentence multiple times, but also number each sentence in sequence, and then uppercase the words in sequence, lowercasing the rest. She figured this would affect more skin and cause more pain and suffering on Harry's part, much to her morbid delight.

Write a program that outputs this sentence **40 times**, numbering each sentence, but also uppercases each word in the sentence, one at a time, in forward sequence, and then in reverse sequence once the final word of the sentence is reached, over and over again, like this:

```
1  I must not tell lies.
2  i MUST not tell lies.
3  i must NOT tell lies.
4  i must not TELL lies.
5  i must not tell LIES.
6  i must not TELL lies.
7  i must NOT tell lies.
8  i MUST not tell lies.
9  I must not tell lies.
10 i MUST not tell lies.
11 i must NOT tell lies.
12 i must not TELL lies.
```

... and so on

Input: None.

Output: The list of forty sentences as described and partially shown above, using the pattern indicated. All sentences must be numbered, and then **aligned exactly as shown**.

2. Bingwen

Program Name: Bingwen.java

Input File: bingwen.dat

Bingwen is calculating third roots of values, has trouble doing it by hand, and doesn't have a scientific calculator with that function, which would certainly make it easy to do. For example, the third root of 8 is 2, which simply means $2 \times 2 \times 2$ equals 8. Can you help him?

Input: Several values, each on one line, each no larger than 2147483647, or no smaller than its complement.

Output: The third root of each value, rounded to two decimal places.

Sample input:

```
8
234
6124
-699
84975
-33402
```

Sample output:

```
2.00
6.16
18.30
-8.87
43.96
-32.21
```

3. Caleb

Program Name: Caleb.java

Input File: caleb.dat

Caleb’s infantry company has been deployed to Bogswamistan for the next two years. Upon arrival Caleb discovers that Bogswamians tell time differently than we do here in the US. Rather than seconds, minutes and hours they simply use degrees as a unit of time. Where once around the face of a clock for us is 60 minutes or 12 hours, once around the face of a Bogswamistanian clock is 360 degrees. Bogswamians simply report the time in two measures of degrees, separated by a slash. For example, in Bogswamian time 12:00 noon for us would be 0/0. 6:00 pm (and am) would be 180/0. 3:45 our time would be 112/270 in their time. Help Caleb write a program that will convert a Bogswamian time to one the rest of his company can understand.

Input: A whole number N representing the number of Bogswamian times to convert to standard time followed by N “degree” times shown as two whole numbers H and M separated by a colon. H and M will both be greater than or equal to 0 and less than 360. None of the Bogswamian times will contain leading zeroes. There will not be a designation for am or pm in either format.

Output: For each of the N Bogswamian times, output a standard US time in hours and minutes. All minute values should be displayed using two digits. Single digit hours should be displayed as a single digit. (Examples: 9:05 10:09 12:15 4:00) A minute in standard time does not elapse until the equivalent degrees in Bogswamian time has elapsed.

Sample input:

```
5
0/0
180/0
112/270
215/60
357/330
```

Sample output:

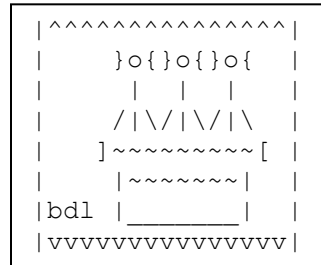
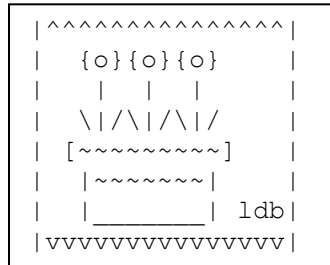
```
12:00
6:00
3:45
7:10
11:55
```

4. Debra

Program Name: Debra.java

Input File: debra.dat

Debra loves ASCII art, and has found a really cool website (<http://www.chris.com/ascii>) that has all kinds of beautiful ASCII graphic pictures, created by using only the characters found on the keyboard. She particularly liked the flowers, and found a cute one she really liked, shown below:



She noticed the initials **ldb** as a part of the picture and after looking through the website realized those belonged to the artist who created the picture. Debra decided to play around with the graphic and wanted to see what it would be like if she reversed it. It wasn't quite the mirror image, but pretty close, and it was certainly a fun exercise. Help her by writing a program to output the mirror image of a given graphic, like the example shown above.

Input: Several sets of data, each consisting of two integers R and C, representing the dimensions of a rectangular graphic, enclosed in a frame like the one shown in the example above, followed by R rows with C characters in each row, containing an ASCII graphic picture. All of the blanks are single spaces, no tabs included.

Output: The mirror image of the ASCII graphic, each output followed by “====”.

Sample input:

```

8 17
| ^^^^^^^^^^^^^^^^^ |
| {o} {o} {o}      |
|   |   |   |       |
| \ | / \ | / \ | /  |
| [ ~ ~ ~ ~ ~ ~ ~ ~ ] |
|   | ~ ~ ~ ~ ~ ~ ~ |
|   | _____ | ldb |
| vvvvvvvvvvvvvvvvvv |
14 18
| ^^^^^^^^^^^^^^^^^ |
|                       |
|                       |
| , ' - . - .         |
|   /// // // )       |
|   /// // // // |   |
|   /// // // //    |
|   /// // // //    |
| ( : // // //       |
| ` ; ` : // //      |
|   / / : ` : /      |
| // // ` '          |
| ( _ / hh           |
| vvvvvvvvvvvvvvvvvv |

```

Sample output:

```

| ^ ^ ^ ^ ^ ^ ^ ^ ^ ^ ^ ^ ^ ^ ^ |
|           } o { } o { } o {      |
|           |       |       |       |
|           / | \ / | \ / | \       |
|           ] ~ ~ ~ ~ ~ ~ ~ ~ [      |
|           | ~ ~ ~ ~ ~ ~ ~ ~ |      |
| bdl | _____ |                 |
| vvvvvvvvvvvvvvvvvvvvvvvvvvvvv |
| ===== |
| ^ ^ ^ ^ ^ ^ ^ ^ ^ ^ ^ ^ ^ ^ ^ |
|           . - . - _                |
|           \ / /  / ' ,              |
|           ) /  //  // //            |
|           | //  //  // //           |
|           ///  //  /// //          |
|           ///  //  /// //          |
|           ///  //  // //           |
|           ///  //  : ` (            |
|           ///  : ` ; `              |
|           / : ` : / /              |
|           ' `      / /             |
|                                   / / |
|                                   hh / _ ( |
| vvvvvvvvvvvvvvvvvvvvvvvvvvvvv |
| ===== |

```

5. Emily

Program Name: Emily.java

Input File: emily.dat

Emily has just learned how to manipulate two-dimensional arrays of values, specifically how to search for patterns hidden within such an array. She wants to practice her skills and has come up with a challenge. Taking a square grid of single digits, Emily decides to do a search for numeric strings inside the grid, like the one shown below.

```
1 2 3 4 5 4
2 5 4 3 2 1
3 5 3 4 5 1
4 4 3 4 5 2
5 3 4 3 4 3
1 4 5 2 3 4
```

Being a very methodical and systematic problem solver, Emily always starts at the top left corner of the grid, making her way across that row, then to the next row all the way across, and so on until she reaches the bottom right of the grid. She does this grid traversal for each of the eight directions she is searching, first looking northwards, then towards the northeast, then east, and on around the remaining directions in clockwise order. It is certainly possible that the search string can be found in multiple places, even being in the same row, same column, or same diagonal. It is also possible that the string is nowhere to be found in the grid. Furthermore, Emily decides that the numeric strings can “wrap around” in any of the eight cardinal directions.

For example, the numeric string 3451 is found in two places, first at position (3,3) going east and then at position (3,1) going south. Another string, 425, is found four times: (1,6) SE, (5,3) SE, (6,2) S, and (4,1) W. A “wrap around” example starts at position (6,2), wrapping around to the top of grid, and also starting at position (1,6) and going diagonally towards the southeast, and then going down a row and all the way to the first column for the 2 digit, and the 5 one more row down and one column to the right.

She is able to do small grid searches by hand but needs your help to write a program for processing larger data sets and longer strings, reporting all found instances first by direction (search northwards first, then clockwise from there), then for each search direction going by row, and finally by column.

Input: An initial N, indicating the size of the square grid of single digits, followed by an NxN grid of single digits ranging from 0 to 9, each row on a separate line, with single space separation between digits. Following the grid will be several numeric strings S, also containing the digits 0 through 9, with no spaces between digits.

Output: For each string S, show the string inside angle brackets as shown below, and report the locations where S is found in the grid, in direction, row and then column order, as described above and shown in the sample output below. If the search string is not found, output the words NOT FOUND.

Sample input:

```
6
1 2 3 4 5 4
2 5 4 3 2 1
3 5 3 4 5 1
4 4 3 4 5 2
5 3 4 3 4 3
1 4 5 2 3 4
3451
425
546
5433
351
```

Sample output:

```
<3451>
(3,3) E
(3,1) S
<425>
(1,6) SE
(5,3) SE
(6,2) S
(4,1) W
<546>
NOT FOUND
<5433>
```

```
(6,3) N
(5,1) NE
(5,1) SE
(3,2) SW
<351>
(5,4) NE
(3,3) NW
```

6. Joe

Program Name: Joe.java

Input File: joe.dat

Joe has discovered a literary classic, a wonderful and poignant story about rabbits, believe it or not, in a quest for survival called **Watership Down**, by Richard Adams, published in 1974. He especially loves the preambles for each chapter, which are quotes from other literary classics. For example, the one for the first chapter is an excerpt from Aeschylus, Agamemnon, where Cassandra and the chorus have an exchange.

Chorus: Why do you cry out thus, unless at some vision of horror?
Cassandra: "The house reeks of death and dripping blood."
Chorus: How so? Tis but the odor of the altar sacrifice.
Cassandra: "The stench is like a breath from the tomb."

These preambles are meant to introduce a particularly strong theme for that chapter, in this case introducing Fiver, a small and troubled rabbit who has visions of impending doom for their home, much like the visions of Cassandra.

Joe decides to use some of these preambles in a simple text processing exercise where he takes all the words of the sentence, uppercases and alphabetizes them by length, eliminating punctuation marks and all duplicates. For example, the result of the lines above would be:

AT DO OF CRY OUT WHY YOU SOME THUS HORROR UNLESS VISION
OF AND THE BLOOD DEATH HOUSE REEKS DRIPPING
OF SO BUT HOW THE TIS ODOR ALTAR SACRIFICE
A IS THE FROM LIKE TOMB BREATH STENCH

Input: Several lines of text as shown in the sample input below. No punctuation symbols are to be included in the resulting words, including periods, commas, question marks, exclamation marks, or quotes. There will be no hyphenated words or contractions in the data.

Output: For each line of text, output all words in alpha order by size, as described above and shown below.

Sample input:

Why do you cry out thus, unless at some vision of horror?
"The house reeks of death and dripping blood."
How so? Tis but the odor of the altar sacrifice.
"The stench is like a breath from the tomb."
The darksome statesman, hung with weights and woe
Like a thick midnight fog, moved there so slow

Sample output:

AT DO OF CRY OUT WHY YOU SOME THUS HORROR UNLESS VISION
OF AND THE BLOOD DEATH HOUSE REEKS DRIPPING
OF SO BUT HOW THE TIS ODOR ALTAR SACRIFICE
A IS THE FROM LIKE TOMB BREATH STENCH
AND THE WOE HUNG WITH WEIGHTS DARKSOME STATESMAN
A SO FOG LIKE SLOW MOVED THERE THICK MIDNIGHT

7. Opus

Program Name: Opus.java

Input File: opus.dat

The Firethorne High School Computer Science Club is sponsoring an invitational computer science competition to raise money to pay for much needed compute time on The University of Texas' super computer Stampede. The invitational will be just like all other UIL competitions in that there will be a programming portion and a written exam. The club sponsor, Mr. Snigglefritz, has asked Opus to help out with grading the written portion of the contest. Each contestants name, name of their school, school's classification (1A, 2A, etc.) and their answers to the 40 written exam questions will be scanned into a file along with the key to the exam. Opus must write a program that will read and score each contestant's answers. It must then print a report that separates the contestants by school classification and ranks them based on their score on the test.

The usual UIL scoring methods will be used. Contestants will earn 6 points for a correct answer, lose 2 points for an incorrect answer and 0 points for any question that is skipped. Ties will be broken using percent correct. The student with the greatest percent correct will win the tie.

Input: The first line of the data file will be the word KEY followed by 40 letters A through E that represent the correct answers to the written exam each separated by a space. Next will be several lines of data each containing the following: first name, last name, school name (all will be one word), school classification and 40 capital letters A through E or an S if the question has been skipped. Each of the data items will be separated by a single space.

Output: A report that shows the results of the competition. The report should be grouped by school classification from 1A to 6A. Within each classification the contestants should be listed in descending order based on their test score each on a separate line. For each contestant show their placing, last name followed by a comma, their first name, their school's name, and their score on the exam. All items should be separated by a single space.

Ties should be broken and displayed in the correct order with correct place. The tie breaker goes to the student with a higher percentage of correct answers. If two students have the same score and same percentage of correct answers, the tie is considered to be unbreakable. Unbreakable ties should be shown as having the same place and in alphabetical order based on last name, then first name. The next place after an unbreakable tie should reflect the same place, not the next place. For example, if there is an unbreakable tie for first place, list both contestants as first and then list the next contestant as third place, not second.

Sample input:

```
KEY A E C B C B D C A E A C E B D E C D A C E C C B C D E E A C A B B B D B C D A D
Sebastian Williams Channing 1A A E C B C E D C E E E C E C E E C E A C E C D A B S S B A C A B E A D B C C E S
Alexander Taylor Gorman 1A A A S B A S D B S E C S E C S E D S A D E C B S C A E B B C A B S B B S C B S D
Jose Rodriguez Shamrock 1A S E C B C B D S A E E B C C D E C C A A A S E B B D E C A S S A E D D E S B A E
Angel Harris Rusk 4A A E D D C E B C C E A C A A B E A B B C E E B C D A A C A A D A B B C C E D A A
Sofia King Odessa 6A A E C B C B C C B E A C E C D B S B D S C D S D C E A C E S D D S B E S D D S S
Aria Morris Frost 2A C C C C C C C C S S S S S S S S S S C C C C C C C C C C C C C C C C C C C C C C
Jacob Miller Zapata 4A A A D E E B B D B B E A B B D E C A E E E C C B A A D C C B C C D E E D C E E B
Michael Martinez Frost 2A A E D B E B D C A E E B D C D E E C A C B C C S C C S E B S D S S S S B C S A S
Emily Adams Alvin 6A A E C B C B D B A E E C E C D E C C A C E C D B C D E C A C A E E B C S S S S S
Isaac Robinson Alvin 6A C E C A D B B C E E E B E B D E C A A D E A D E C A E C A D A A E E D C C A A A
Dylan Lopez Tulia 3A C A A B E E A B B C E D D B D B C C E D C E B D C B D D A B E E E D A D C B B D
Camila Green Poolville 2A A E C A C B D C A D E C E C D E C C A C E C D B C D S B S S A E E B D B S E A D
Penelope Stewart Van 4A D E C B C B D C C B E C E C D E C D E C D E C D E D A C A S E B S B C E A D
David Wilson Rusk 4A A E C B C B D C A E E C E C E E C C A C B D B E S C C E B A S C B E S E C S E C
Scarlett Turner Junction 2A A E D B E B D C A E E B D C D E E C A C B C C S C C S E B S D S S S S B C S A S
Jayden Jackson Poolville 2A A E C B C B D C A E E C E C D E C C A C E E A D E A D E D D C C D D B C C D B S
Elijah Davis Lasara 2A C E C B C B E C A E E C E C E E C E A C S C D S C D E E A C A B S S C D C B E A
Abigail Baker Lamesa 4A C E C C B E B A E E C E D E S B C S S S S S C C S S S S S S S S S S S S S S S
Natalie Flores Marfa 1A A E C B C B D C A E E C E C D S S S S S S S S S S S S A A A A A S S S S S S S S
Luke White Poolville 2A A E C B C B D A A E E C E C D S S S S S S S S S S S S S S S S S E E E E S S S S S
James Martin Hooks 3A A E C B C D D C C E E C E A D S B S S S S E S S S S D E S S S S C E E S S S S S D
```

Opus - continued

Sample output:

1A

1 Williams, Sebastian - Channing: 110
2 Flores, Natalie - Marfa: 80
3 Taylor, Alexander - Gorman: 76
4 Rodriguez, Jose - Shamrock: 76

2A

1 Green, Camila - Poolville: 136
2 Davis, Elijah - Lasara: 120
3 Martinez, Michael - Frost: 90
3 Turner, Scarlett - Junction: 90
5 Jackson, Jayden - Poolville: 90
6 White, Luke - Poolville: 58
7 Morris, Aria - Frost: 12

3A

1 Martin, James - Hooks: 74
2 Lopez, Dylan - Tulia: -16

4A

1 Stewart, Penelope - Van: 140
2 Wilson, David - Rusk: 80
3 Harris, Angel - Rusk: 40
4 Baker, Abigail - Lamesa: 16
5 Miller, Jacob - Zapata: 8

6A

1 Adams, Emily - Alvin: 138
2 Robinson, Isaac - Alvin: 72
3 King, Sofia - Odessa: 56

8. Prashant

Program Name: Prashant.java

Input File: prashant.dat

Prashant has just learned about permutations and knows how to calculate how many possible unique permutations can be generated from a simple list of items. For example, he understands that given three elements A, B and C, there are six different ways these three elements can be listed in order: ABC, ACB, BAC, BCA, CAB, and CBA. A way to express the total is to use the factorial function in mathematics, expressed as $3!$, 3 being the number of elements, and the ! operator indicating a “count down and multiply” process, like this:

$$3 \times 2 \times 1 = 6$$

Furthermore he learned that if there are duplicates, like in the set of elements A, B, C and C, there are indeed $4!$, or 24 permutations, but since C is duplicated, there are only half as many unique permutations, which would be 12:

ABCC, ACBC, ACCB, BACC, BCAC, BCCA, CABC, CACB, CBAC, CBCA, CCAB, CCBA

The general expression to reduce the value due to duplicates would be $\text{num!}/\text{dup!}$, where **num** is the number of elements, and **dup** is the number of duplicates. This formula would still work if the value of **dup** was zero (indicating no duplicates), resulting in $0!$, or 1, since by definition, $0!$ is equal to 1. For the two examples so far, the calculations would be:

$$\begin{aligned} 3!/0! &= 6 \\ 4!/2! &= 12 \end{aligned}$$

However, things got a bit more interesting when he tried the sequence A B A B, which had **two** elements duplicated. He discovered that he needed to alter the denominator of the formula, accounting for the multiple duplicates, resulting in this formula, $\text{num!}/(\text{dup1!} * \text{dup2!})$, which would produce the expression:

$$4!/(2!*2!) = 6$$

With even more complex sequences, for each element that was duplicated, he had to add another factor to the denominator to account for each additional duplication. For instance, the sequence A B C A B C D A has three elements that are duplicated, one even occurring three times, requiring a denominator of $3!*2!*2!$. The resulting number of unique permutations for this list is 630.

Prashant realized this was getting quite complicated and decided to reach out to you for help. He is only interested in calculating the number of **unique permutations** that would be generated, given a list of elements, and not the actual permutations. Can you help him write a program to do this?

Input: Several lines of data, each containing a list of elements represented by single uppercase letters, with single space separation, each sequence possibly containing duplicate elements as demonstrated above. There will be no more than 15 elements in each line of data.

Output: The total number of unique permutations that would be generated with the given data, eliminating any duplicate permutations due to duplicate elements.

Sample input:

```
A B C
A B C C
A B A B
A B C A B C D
```

Sample output:

```
6
12
6
630
```


9. Ramya

Program Name: Ramya.java

Input File: ramya.dat

Ramya's friend Prashant has been researching permutations, specifically calculating the number of unique possible sequences that can be generated from a list of elements. However, Ramya wants to actually generate the list of unique permutations, which she considers much more of a research challenge than just counting how many there could be, as Prashant is doing.

She knows that given three elements A, B and C, there are six different ways these three elements can be listed in order, starting with the original string, and proceeding from there as follows, ABC, ACB, BAC, BCA, CAB, and CBA, a total of six unique sequences.

She also knows that if there are duplicate elements, as in the list A, B, C and C, there are duplicate sequences, which she wants to eliminate. The unique list would be as shown below.

ABCC, ACBC, ACCB, BACC, BCAC, BCCA, CBAC, CBCA, CABC, CACB, CCAB, CCBA

From Prashant's research, she also knows that with multiple duplicates in the list, the process becomes even more complicated, and with more and more elements, the list of unique permutations can become quite long. She decides that she will stop generating permutations once the list reaches 30 permutations, even though there could be many more to follow. She needs your help to write the program to generate up to the first 30 unique permutations of a list of elements, in the order pattern demonstrated in all the examples shown on this page, starting with the original sequence.

Input: Several lines of data, each containing a list of elements represented by single uppercase letters, with single space separation, each sequence possibly containing duplicate elements as demonstrated above. There will be no more than 7 elements in each line of data.

Output: The numbered list of unique permutations for each given list, in the exact order pattern and format shown below, each sequence ending with the single line "=====".

Sample input:	BCCA	BACCC
ABC	CBAC	BCACC
ABCC	CBCA	BCCAC
ABAB	CABC	BCCCA
ABCCC	CACB	CBACC
	CCAB	CBCAC
	CCBA	CBCCA
Sample output:	=====	CABCC
ABC	ABAB	CACBC
ACB	ABBA	CACCB
BAC	AABB	CCABC
BCA	BAAB	CCACB
CBA	BABA	CCBAC
CAB	BBAA	CCBCA
=====	=====	CCCBA
ABCC	ABCCC	CCCB
ACBC	ACBCC	=====
ACCB	ACCCB	
BACC	ACCCB	
BCAC	ACCCB	

10. Samantha

Program Name: Samantha.java

Input File: samantha.dat

In accounting class, Samantha is learning how to do simple calculations of subtotal, tax and total, using 8.25% for the tax. She is required to take a list of values, add them up, and output these three values in a column format. For now she will only deal with total values of less than \$10,000. Can you help her by writing a program to do this to help her check her work?

For example, a list of values like this:

9.99 10.53 45.87 100 613.25

would have a final result of this:

**Subtotal: \$ 779.64
Tax : \$ 64.32
Total : \$ 843.96**

Input: Several lines of values, each set of values on one line, with single space separation.

Output: The accounting result of the line of values, as described and shown above, as well as in the samples below. Each complete output is to be ended with "=====

Sample input:

9.99 10.53 45.87 100 613.25
12 13 14 15 16 17 18

Sample output:

Subtotal: \$ 779.64
Tax : \$ 64.32
Total : \$ 843.96
=====

Subtotal: \$ 105.00
Tax : \$ 8.66
Total : \$ 113.66
=====

11. Vincent

Program Name: Vincent.java

Input File: vincent.dat

Vincent was born on April 19, 2002 at 5:45 AM. Recently he has been thinking what the new date and time would be after a certain number of minutes had elapsed since that original time. For example, it was easy for him to think about a relatively small number of minutes, like 60 or 420 or even 1440 minutes later, in essence an hour, seven hours and 24 hours later.

He decided to represent his birth data as follows, and then list the number of elapsed minutes on the next line, like this:

```
2002 4 19 5 45
60 420 1440
```

He also decided on the date/time output format shown below to represent his birth date, followed by the other dates and times he thought about:

```
April 19, 2002 - 05:45 AM
-----
April 19, 2002 - 06:45 AM
April 19, 2002 - 12:45 PM
April 20, 2002 - 05:45 AM
```

He needs your help to write a program that will do this, given any starting date and time, followed by any number of minute values representing number of minutes elapsed from then. **Be sure to account for any leap year situations.**

Input: Several sets of data, each set occupying two lines, the first of which is the original date/time in the format:

yyyy mm dd hh mm

followed by a second line with several integer values, single space separation, representing elapsed minutes since the original time. He didn't consider any year before 1970 in his test data, represented hours in the range 0-23, and minutes from 0-59. The month and day numbers matched the date intended, with months ranging from 1-12, and day numbers ranging according to that month.

Output: The original date formatted as shown in the example above, followed on the next line by "-----", and then on the next several lines by the dates and times that represent the elapsed minute values in the second data line, ending with "=====" to end the output for that data set.

Sample input:

```
2002 4 19 5 45
60 360 1440 380 1095 4621
1990 8 6 17 39
2526861 81741 2368706 48621
```

Sample output:

```
April 19, 2002 - 05:45 AM
-----
April 19, 2002 - 06:45 AM
April 19, 2002 - 11:45 AM
```

```
April 20, 2002 - 05:45 AM
April 19, 2002 - 12:05 PM
April 20, 2002 - 12:00 AM
April 22, 2002 - 10:46 AM
=====
August 06, 1990 - 05:39 PM
-----
May 27, 1995 - 12:00 PM
October 02, 1990 - 12:00 PM
February 06, 1995 - 03:05 PM
September 09, 1990 - 12:00 PM
=====
```

12. William

Program Name: William.java

Input File: william.dat

William is fascinated with palindromes, and knows many of the classic ones, like “**Madam, I’m Adam**”, “**Rise to vote, sir.**”, and “**A man, a plan, a canal, Panama**”, but is bored with those and is now looking at a new challenge, palindromes involving sequences of numeral digits, like 61052739404563204519876054312614. He clearly sees that this list of digits is not a palindrome but wonders if there is one hiding inside. He decides to divide the sequence into two parts, like this: 6105273940456320 4519876054312614.

He immediately sees a 61 in the first part matched by a 16 in the second part, which would be considered an imbedded palindrome, written like this, “61:16”. He decides to look further and see if there was one that was longer, and sure enough, he sees “613:136”, and then a longer one, “6134:4136”. He keeps searching and finally decides that the largest embedded palindrome he can find is “123450:054321”.

For curiosity’s sake, he takes the first sequence, chops off one digit at the end, and tries it again. When he splits up the digits this time he gets 610527394045632 0451987605431261, the first list one character shorter than the last. He is surprised that the resulting palindrome - 6157940:0497516 - is actually longer than the first example, which he finds very interesting.

This searching process took a lot of trial and error, and William could use your help in finding a better way to do this, perhaps a computer program that does all the work for him.

Input: Several lines of data, each containing a list of contiguous digits.

Output: The largest “palindrome” contained in the list, printed in the format described above and shown below. If no palindrome exists, output a single colon with no values on either side.

Sample input:

```
61052739404563204519876054312614
6105273940456320451987605431261
883625174635
123456789
```

Sample output:

```
123450:054321
6157940:0497516
36:63
:
```