UIL COMPUTER SCIENCE WRITTEN TEST

2018 DISTRICT

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 DISTRICT

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 the sum of 11111111_2 and 11110011_2?

A) $1F3_{16}$ **B)** 497_{10} **C)** 762_8 **D)** 111010010_2 **E)** None of the above.

Question 2.

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

A) `18` **B)** `18.0` **C)** `10.5` **D)** `10` **E)** `11`

```
out.println(10+5-3*6.0/4);
```

Question 3.

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

A)
```
Go

Spurs
Go!
```

B)
```
Go
Spurs
Go!
```

C)
```
Go
Spurs Go!
```

D)
```
Go Spurs
Go!
```

E) `Go Spurs Go!`

```
out.println("Go\n");
out.print("Spurs\nGo!");
```

Question 4.

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

A) `laoon` **B)** `ploon` **C)** `lmoon`

D) `plmoon` **E)** `loon`

```
String s="planet";
String t="moon";
String u=s.substring(1, 2)+t.substring(1);
out.print(u);
```

Question 5.

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

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

```
out.print(true^false&&true||false);
```

Question 6.

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

A) `4` **B)** `4.0` **C)** `8` **D)** `8.0` **E)** `2`

```
int x=64;
out.print(Math.cbrt(x));
```

Question 7.

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

A) `68.0` **B)** `D` **C)** `68` **D)** `68.35`

E) Will not compile. Type mismatch error.

```
char c='A';
int i=8;
double d=4.65;
out.print(c+i-d);
```

Question 8.

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

A) `yes`

B) `no`

C) `yes and no`

D) `maybe`

E) `yes yes and no`

```
boolean yes=false,no=true,maybe=true;
if(yes)
  out.print("no ");
else if(no)
  out.print("yes ");
else if(maybe)
  out.print("yes and no ");
else
  out.print("maybe");
```

Question 9.

Which of the following must replace **<code>** in the loop shown on the right to ensure that the code segment will print exactly 6 X's?

A) `int i=1;i<10;i+=2`

B) `int i=0;i<=10;i+=2`

C) `int i=1;i<6;i++`

D) `int i=0;i<=6;i++`

E) `int i=1;i<=6;i+=2`

```
for(<code>)
     out.print("X");
```

Question 10.

What is output by the code segment listed to the right?

A) `[0, 8, 12, 10, 0]`

B) `[8, 12, 10, 12]`

C) `[0, 8, 12, 10, 12]`

D) `[8, 12, 10, 12, 0]`

E) Error. Throws an ArrayIndexOutOfBoundsException.

```
int[] list= new int[5];
list[1]=8;
list[2]=12;
list[3]=10;
list[4]=list[list[2]-list[3]];
out.print(Arrays.toString(list));
```

Question 11.

What is printed by the code segment shown on the right if the following values are contained in `datafile.dat`? Assume that all necessary classes have been imported and that the main method throws an IOException.

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

A) `16 -3`

B) `16 4`

C) `22 -3`

D) `22 4`

E) Error. Throws a NoSuchElementException .

```
File f=new File("datafile.dat");
Scanner s=new Scanner(f);
int a=0;
while(s.nextInt()>0)
     a+=s.nextInt();
out.print(a+" "+s.nextInt());
```

Question 12.

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

A) `44.0 1`

B) `55.0 1`

C) `45.0 1`

D) `55.0 0`

E) `45.0 0`

```
double d=0;
int i=10;
do {
     d+=--i;
}while(i>0);
out.print(d+" "+i);
```

Question 13.

In any given expression, which of the following operators would be applied last?

A) `&&` **B)** `*` **C)** `<=` **D)** `^` **E)** `||`

Question 14.

Which of the following statements will not compile?

A) `long l=Short.MAX_VALUE;`

B) `int i=Byte.BYTES;`

C) `int j=Byte.SIZE;`

D) `byte b=Integer.MIN_VALUE;`

E) `short s=Byte.MAX_VALUE;`

Question 15.

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

A) `[6, 0, 4, 5]`

B) `[6, 4, 5]`

C) `[6]`

D) `[0, 4, 5]`

E) `[5]`

```
ArrayList<Integer> a=new
ArrayList<Integer>();
a.add(4);
a.set(0, 0);
a.add(5);
a.set(0, 6);
a.remove(1);
out.print(a);
```

Question 16.

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

A) `four three two two one`

B) `four three two one`

C) `one two three four`

D) `one two two three four`

E) `four three one`

```
Stack<String> s=new Stack<String>();
s.push("one");
s.push("two");
s.push("two");
s.pop();
s.push("three");
s.push("four");
while(!s.isEmpty())
     out.print(s.pop()+" ");
```

Question 17.

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

A) `PecosPecoPecPe`

B) `PPePecPecoPecos`

C) `PecosPecoPecPeP`

D) `PePcePocePsoceP`

E) `PPPPP`

```
public static String rec(String s,int i) {
if(s.length()==1)
     return s;
else
     return s+rec(s.substring(0,i),i-1);
}
//client code
String s="Pecos";
out.print(rec(s,s.length()-1));
```

Question 18.

Which of the following should replace **<code 1>** in the class shown on the right?

A) `double`

B) `int`

C) `static`

D) `this`

E) `super`

Question 19.

Which of the following should replace **<code 2>** in the class shown on the right?

A) `l,w,h`

B) `double length,double width,double height`

C) `length,width,height`

D) `double l,double w,double h`

E) No additional code is required

Question 20.

What is the output if this client code that is implemented in a different class than `Box`.

```
Box b1=new Box(10,10,10);
out.print("Height="+b1.height+" ");
out.print(b1.surfaceArea()+" ");
out.print(b1.volume);
```

A) `10.0 600.0 1000.0`

B) `Height=10.0 300.0 1000.0`

C) `Height=10.0 600.0 1000.0`

D) `Height=10.0 1000.0 600.0`

E) There is no output due to an error.

```
//Use the following code to answer questions
//18, 19 and 20.

public class Box {

public <code 1> surfaceArea() {
  return 2*(height*width+length*
            height+length*width);
  }

public Box(<code 2>) {
  length=l;
  width=w;
  height=h;
  volume=length*width*height;
}

private double length,width,height;
public double volume;
}
```

Question 21.

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

A) `[4, 5, 6, 7]`

B) `[2, 3, 4, 5]`

C) `[2, 4, 6, 8]`

D) `[1, 3, 5, 7]`

E) `[6, 7, 8, 9]`

```
int[][] mat= new int[4][4];
for(int x=0;x<4;x++)
      for(int y=0;y<4;y++)
            mat[y][x]=x+2*y;
out.println(Arrays.toString(mat[2]));
```

Question 22.

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

A) `true true false`

B) `true false true`

C) `false true true`

D) `false false true`

E) `false true false`

```
out.print("123ABC".matches("\\D{3}\\W{3}")+" ");
out.print("555-5555".matches(".+")+" ");
out.print("Alphabet".matches("A[a-z]?"));
```

Question 23.

Which of the following represents the correct signature of a method named `tip` that has an amount for a meal and the desired tip percent as its parameters and returns the appropriate tip amount?

A) `public static void tip(double amount, int percent)`

B) `public static tip(double amount, int percent)`

C) `public static double tip(amount, percent)`

D) `public static double tip(double amount, int percent)`

E) `tip(double amount, int percent)`

Question 24.

Which of the following methods will return N! (N factorial) ?

A)

```
public static long fac(long n) {
long f=1;
for(long x=n;x>=1;x--)
     f*=x;
return f;
}
```

B)

```
public static long fac(long n) {
long f=1,x=2;
while(x<=n) {
     f=f*x;
     x++;
}
return f;
}
```

C)

```
public static long fac(long n) {
if(n==1)
     return 1;
else
     return fac(n-1);
}
```

D) A and B

E) A, B and C

Question 25.

Which of the following Java expressions will correctly round `n` to `r` decimal places if `n` is a `double` and `r` is an `int`?

A) `(int)(r*Math.pow(10, n)+0.5)/Math.pow(10, n)`

B) `(n*Math.pow(10, r)+0.5)/Math.pow(10, r)`

C) `(int)(n*Math.pow(10, r)+0.5)/Math.pow(10, r)`

D) `(int)(n*10+0.5)/10`

E) `(int)(n/Math.pow(10, r)+0.5)*Math.pow(10, r)`

Question 26.

What is the smallest possible value that the code shown on the right will produce?

A) 6

B) 11

C) 66

D) 1

E) 0

```
Random r=new Random();
System.out.print(r.nextInt(6)*11);
```

Question 27.

Which of the following must replace **<code>** in the method shown on the right to ensure the method will sort `a` in ascending order?

A) `k>=0&&a[k]<ce`

B) `k>=0&&a[k]>ce`

C) `k>=0||a[k]>ce`

D) `k>=i&&a[k]>ce`

E) `k>=ce&&a[i]>ce`

Question 28.

Once **<code>** has been filled in correctly, which of the following sorting algorithms is implemented by the `uilSort` method?

A) heap sort

B) quick sort

C) insertion sort

D) selection sort

E) merge sort

Question 29.

What is the least restrictive worst case time efficiency (Big O value) for the `uilSort` method?

A) O(1)

B) O(n)

C) O(n^2)

D) O(log n)

E) O(n log n)

Question 30.

Which of the following shows the order of the elements in array `a` when code execution reaches the comment statement and `i` equals 2 given the following client code?

```
int[] a= {5,3,1,0,2,4};
uilSort(a);
```

A) `[0, 1, 3, 5, 2, 4]`

B) `[1, 2, 3, 0, 5, 4]`

C) `[5, 3, 1, 4, 2, 0]`

D) `[1, 3, 5, 0, 2, 4]`

E) `[0, 1, 2, 5, 3, 4]`

```
//Use the following method to answer
//questions 27, 28, 29 and 30.
public static void uilSort(int[] a) {
  int i=1;
  do {
    int ce=a[i];
    int k=i-1;
    while(<code>) {
      a[k+1]=a[k];
      k--;
      }
    a[k+1]=ce;
    //comment
    i++;
  }while(i<a.length);
}
```

Question 31.

What is the output of the code segment shown here given the method implementation on the right?

```
int g,h=0;
for(g=1;g<=3;g++)
  out.print(doSomething(g,h)+" ");
out.print(g+" "+h);
```

A) 13 14 15 3 0

B) 13 14 15 4 0

C) 22 23 24 4 0

D) 22 23 24 9 6

E) 13 14 15 9 6

```
public static int doSomething(int g,int h) {
while(h<=5) {
      g=h+++g;
      h++;
      }
return g+h;
}
```

Question 32.

What is printed by the line of code shown on the right?

A) 14

B) 0

C) 30

D) 15

E) 16

```
out.print(14|15&16);
```

Question 33.

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

A) Go

B) Fight

C) Win

D) Error. Will not compile.

E) Error. Throws a run time exception.

```
Double d1=new Double(18.99);
Double d2=19.00;
if(d1.compareTo(d2)==0)
     out.print("Go");
else if(d1.compareTo(d2)>0)
     out.print("Fight");
else
     out.print("Win");
```

Question 34.

Which of the following lines of code will not compile correctly?

A) int i=2147483647;

B) double d=250.84d;

C) int h=0xABC;

D) char c=0b11111111;

E) None of the above. All of the lines shown above will compile correctly.

Question 35.

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

A) #@&*%@@%&&

B) #@&*@@%&&

C) #@@*%@@%@&

D) #@@*@@%@&

E) #&*%%&&

```
String s="March2018",t="";
for(int i=0;i<s.length();i++) {
      switch(s.substring(i, i+1)) {
      case "M":t+="#";break;
      case "c":t+="*";
      case "0":t+="%";break;
      case "r":
      case "1":
      case "8":t+="&";break;
      default:t+="@";
      }
}
out.print(t);
```

Question 36.

Which pair of the Boolean expressions listed on the right are equivalent?

A) I and II

B) II and III

C) III and IV

D) I an IV

E) II and IV

I. $\bar{A} * \bar{B}$

II. $\overline{A * B}$

III. $\bar{A} + \bar{B}$

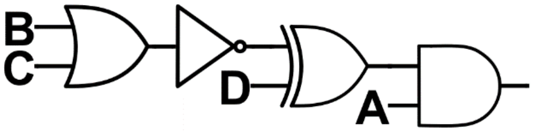
IV. $\overline{A \oplus B}$

Question 37.

What is the value of the Boolean expression shown in the diagram on the right if A is true, B is false, C is true and D is false?

A) true

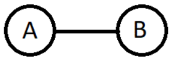
B) false



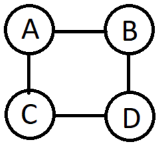
Question 38.

Which of the graphs illustrated here is a complete graph?

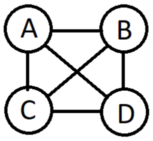
A)



B)



C)



D) A and C

E) A, B and C

Question 39.

Evaluate the prefix expression shown on the right and write your answer in the blank provided?

* − + 8 5 3 2

Question 40.

What is the decimal equivalent of this signed binary 8-bit two's complement value?

10101010

★ANSWER KEY – CONFIDENTIAL★

UIL Computer Science – 2018 District

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

1) C

2) C

3) A

4) E

5) A

6) B

7) D

8) A

9) B

10) C

11) B

12) E

13) E

14) D

15) C

16) B

17) C

18) A

19) D

20) E

21) A

22) E

23) D

24) D

25) C

26) E

27) B

28) C

29) C

30) D

31) B

32) A

33) C

34) E

35) A

36) B

37) B

38) D

*39) 20

*40) –86

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

Note: Correct responses are based on **Java SE Development Kit 8 (JDK 8)** from Sun Microsystems, Inc. All provided code segments are intended to be syntactically correct, unless otherwise stated (e.g., "error" is an answer choice) and any necessary Java SE 8 Standard Packages have been imported. Ignore any typographical errors and assume any undefined variables are defined as used.

Explanations:

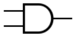
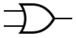
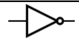
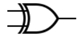
<table>
<tr><td>1.</td><td>C</td><td>11111111_2+11110011_2=111110010_2 (eliminates D). 111110010_2 = 498_{10} (eliminates B). $1F3_{16}$ = 499_{10} (eliminates A). 762_8 = 498_{10}.</td></tr>
<tr><td>2.</td><td>C</td><td>10+5-3*6.0/4=
10+5-18.0/4=
10+5-4.5=
15-4.5=
10.5</td></tr>
<tr><td>3.</td><td>A</td><td>println method inserts a new line after the string is printed. The \n escape sequence inserts a new line where ever it has been inserted in the string.</td></tr>
<tr><td>4.</td><td>E</td><td><table><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td></td><td>0</td><td>1</td><td>2</td><td>3</td></tr><tr><td>p</td><td>l</td><td>a</td><td>n</td><td>e</td><td>t</td><td></td><td>m</td><td>o</td><td>o</td><td>n</td></tr></table>The two argument substring method starts at the index number of the first argument and goes to the second argument minus one. So, the first substring is from one to one (just the l). The one argument substring method starts at the index number of the argument and continues to the end of the string. In this case 1 to 3 (oon).</td></tr>
<tr><td>5.</td><td>A</td><td>T^F&&T||F=
T&&T||F=
T||F=
T</td></tr>
<tr><td>6.</td><td>B</td><td>The cbrt method returns the cube root of its argument as a double. 4X4X4=64.</td></tr>
<tr><td>7.</td><td>D</td><td>ASCII value of ‘A’ is 65. 65+8-4.65=68.35</td></tr>
<tr><td>8.</td><td>A</td><td>Once a true value is encountered, in this case the boolean variable no is true, the code for that if statement is executed and the remaining else statements are skipped.</td></tr>
<tr><td>9.</td><td>B</td><td>i takes the values 0, 2, 4, 6, 8, and 10. Once I becomes 12, the loop stops. That makes 6 six iterations of the loop.</td></tr>
<tr><td>10.</td><td>C</td><td><table><tr><td>int[] list=new int[5]</td><td><b>0</b></td><td><b>1</b></td><td><b>2</b></td><td><b>3</b></td><td><b>4</b></td></tr><tr><td></td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>list[1]=8</td><td>0</td><td>8</td><td>0</td><td>0</td><td>0</td></tr><tr><td>list[2]=12</td><td>0</td><td>8</td><td>12</td><td>0</td><td>0</td></tr><tr><td>list[3]=10</td><td>0</td><td>8</td><td>12</td><td>10</td><td>0</td></tr><tr><td>list[4]=list[list[2]-list[3]];</td><td>0</td><td>8</td><td>12</td><td>10</td><td>12</td></tr></table></td></tr>
<tr><td>11.</td><td>B</td><td>nextInt returns the next value beyond (in front of) the cursor (pointer) in the datafile and then advances the cursor to the next position, even in the condition for a while loop. 5, 1 and -3 are used in the while loop condition statement. 9 and 7 are added to a. 4 is returned by the final call to nextInt because even though the loop terminates when -3 is read, the cursor advances to the next position.</td></tr>
<tr><td>12.</td><td>E</td><td>i is decremented before it is added to d with each iteration of the loop.
   d i
 9.0 9
17.0 8
24.0 7
30.0 6
35.0 5
39.0 4
42.0 3
44.0 2
45.0 1
45.0 0</td></tr>
<tr><td>13.</td><td>E</td><td>Precedence from first to last for the operators shown is: *   <=   ^   && ||</td></tr>
<tr><td>14.</td><td>D</td><td>The MIN_VALUE for Integer is -2147483648. The largest negative value that can be stored in a variable of type byte is -128.</td></tr>
</table>

<table>
<tr><td>15.</td><td>C</td><td><table>
<tr><td><code>a.add(4);</code></td><td>[4]</td></tr>
<tr><td><code>a.set(0,0);</code></td><td>[0]</td></tr>
<tr><td><code>a.add(5);</code></td><td>[0, 5]</td></tr>
<tr><td><code>a.set(0,6);</code></td><td>[6, 5]</td></tr>
<tr><td><code>a.remove(1);</code></td><td>[6]</td></tr>
</table></td></tr>
<tr><td>16.</td><td>B</td><td><table>
<tr><td colspan="2">Stacks use a first in, last out protocol for accessing data.</td></tr>
<tr><td><code>s.push(“one”);</code></td><td>one</td></tr>
<tr><td><code>s.push(“two”);</code></td><td>one two</td></tr>
<tr><td><code>s.push(“two”);</code></td><td>one two two</td></tr>
<tr><td><code>s.pop();</code></td><td>one two</td></tr>
<tr><td><code>s.push(“three”);</code></td><td>one two three</td></tr>
<tr><td><code>s.push(“four”);</code></td><td>one two three four</td></tr>
<tr><td colspan="2">Elements are popped out from right to left.</td></tr>
</table></td></tr>
<tr><td>17.</td><td>C</td><td><table>
<tr><td>i</td><td>s</td></tr>
<tr><td>5</td><td>Pecos</td></tr>
<tr><td>4</td><td>Peco</td></tr>
<tr><td>3</td><td>Pec</td></tr>
<tr><td>2</td><td>Pe</td></tr>
<tr><td>1</td><td>P</td></tr>
</table></td></tr>
<tr><td>18.</td><td>A</td><td>Since <code>height</code>, <code>width</code> and <code>length</code> are all doubles, <code>surfaceArea</code> must return a double value.</td></tr>
<tr><td>19.</td><td>D</td><td><code>l</code>, <code>w</code>, and <code>h</code> have not been declared locally so they must be passed as parameters. The parameter list must show the type and name of each parameter.</td></tr>
<tr><td>20.</td><td>E</td><td>The field <code>height</code> has been declared private, therefore, it cannot be directly accessed from client code that is in another class than <code>Box</code>.</td></tr>
<tr><td>21.</td><td>A</td><td>The matrix looks like this after the loops are done:
[0, 1, 2, 3]
[2, 3, 4, 5]
[4, 5, 6, 7]
[6, 7, 8, 9]
mat[2] is the third row down.</td></tr>
<tr><td>22.</td><td>E</td><td>\D{3} matches exactly 3 non-digits and \W{3} matches exactly 3 non-word characters.
.+ matches any character one or more times.
A[a-z]? matches a capital A followed by any lower case letter once or not at all.</td></tr>
<tr><td>23.</td><td>D</td><td>A method signature must contain a return type, name and parameter list if necessary. All parameters must have a type and name.</td></tr>
<tr><td>24.</td><td>D</td><td>A and B are correct. For answer choice C to be correct the last line should be:
<code>return n*fac(n-1).</code></td></tr>
<tr><td>25.</td><td>C</td><td>Example where n=4.192837 and r=3.
(int)(4.192837*Math(10,3)+0.5)/Math.pow(10,3)=
(int)(4.192837*1000.0+0.5)/1000.0=
(int)(4192.837+0.5)/1000.0=
(int)4193.337/1000.0=
4193/1000.0=
4.193</td></tr>
<tr><td>26.</td><td>E</td><td><code>nextInt(6)</code> will return a whole number between 0 (inclusive) and 6 (exclusive).
0 * 11 = 0.</td></tr>
</table>

27.	B	This is an insertion sort so we are getting the next element in the unsorted portion of the array then shifting elements to the right until we find the proper place for the unsorted element or when we reach the front of the array. Then the unsorted element is placed (inserted) into the proper location.
28.	C	See #29.
29.	C	Best Case O(n), Average Case O(n^2), Worst Case O(n^2)

30.	D	i=1 [3, 5, 1, 0, 2, 4]
		i=2 [1, 3, 5, 0, 2, 4]
		i=3 [0, 1, 3, 5, 2, 4]
		i=4 [0, 1, 2, 3, 5, 4]
		i=5 [0, 1, 2, 3, 4, 5]

31.	B	The variables `g` and `h` in the client code are unchanged by the calls to the `doSomething` method so their final values are 4 and 0. Within the method, for this expression, `h+++g`, the increment operator is applied to the variable h like this: (h++)+g. Here is a trace of the variable values when the code has been run.
		g=1 h=2
		g=3 h=4
		g=7 h=6
		13
		g=2 h=2
		g=4 h=4
		g=8 h=6
		14
		g=3 h=2
		g=5 h=4
		g=9 h=6
		15

32.	A	14 = 1110_2 15 = 1111_2 16 = 10000_2 01111 & 10000 = 00000 00000 | 1110 = 1110 1110_2 = 14
33.	C	`compareTo` returns 0 if d1 and d2 are equal, a value less than 0 if d1 is less than d2, and a value greater than 0 if this d1 is greater than d2. `Double d2=19.00;` is allowed due to autoboxing.
34.	E	2147483647 is within the range of values for the int data type. The letter d following 250.84 designates the value as a double. It is optional. 0x designates a value as hexadecimal. Hexadecimals can be assigned to int type variables. 0b designates a binary number. 11111111 = 255. 255 is a valid ASCII value.
35.	A	When there is no break statement present execution of the code goes to the next case selector. When there is no code present after a case selector, execution goes to the next case selector.
36.	B	DeMorgan's Law states $\overline{A * B} = \bar{A} + \bar{B}$
37.	B	 is AND.  is OR.  is NOT.  is XOR.
38.	D	Every pair of vertices are connected by an edge in a complete graph. A and D and D and B are not connected in answer choice B.
39.	20	* - + 8 5 3 2 = * - 13 3 2 = * 10 2 = 20
40.	-86	Take the complement of 10101010 to get 01010101 then add 1 to get 01010110 which is 86. We know the answer is negative since the sign bit was one so the final answer is -86.

Computer Science Competition
District 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	Alice
Problem 2	Bayani
Problem 3	Candela
Problem 4	Carla
Problem 5	Diya
Problem 6	Gleb
Problem 7	Jeremy
Problem 8	Kinga
Problem 9	Layla
Problem 10	Max
Problem 11	Nandita
Problem 12	Raymond

1. Alice

Program Name: Alice.java Input File: none

Alice is waving to you from her sailboat! Write a program to output this image, exactly as you see it.

Input: None.

Output: A picture of Alice in her sailboat, having fun out on the water.

Sample input:
None

Sample output:

```
           &
           &&
           &&&
           &&-&
           &&--&
           &&---&
           &&----&
           &&--.--&
           &&--..--&
           &&--...--&
           &&--....--&
           &&--.....--&
           &&--......--&
           &&--.......--&
           &&--........--&
           &&--.........--&
           &&--..........--&
           &&--...........--&
           &&--............--&
           &&--.............--&
            ||                   \o/
            ||                    |
 ======================================
   ==================================
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

2. Bayani

Program Name: Bayani.java Input File: bayani.dat

Bayani wants to printout a report of his bill expenses for the month so far and needs a simple program to do that.

Input: A list of bill expenses, each on one line of a data file, each value greater than zero and less than $1,000.00.

Output: Given a list of values, generate a listing of each value aligned and formatted exactly as shown below, and a final total at the end, with **exactly four spaces** to be allocated for the whole number portion after the $ sign for each value (no commas):

$####.##

Assumption: The final total is guaranteed to fit within the format shown above.

Sample input:

```
6.99
12.87
5.44
99.99
115.87
```

Sample output:

```
        $   6.99
        $  12.87
        $   5.44
        $  99.99
        $ 115.87
Total = $ 241.16
```

3. Candela

Program Name: Candela.java **Input File: candela.dat**

Candela's teacher, who gives very difficult tests, has announced one for next week, and has provided her class some preview information about the questions that will be on the test so that she and her classmates can strategize about how best to approach it. The information provided by the teacher only includes the question number, how difficult the question will be on a scale of 1 to 10, and how many points it will be worth if it is answered correctly, on a scale of 1 to 20. The worth of each question does not necessarily match its difficulty, so it is possible that an easy question will be worth as much if not more than a difficult question. The total number of points available will exceed the maximum score of 100 that will be awarded, so students do not have to answer every question, and will be awarded a score of 100 if they meet it exactly, or even exceed it.

What Candela and her classmates have done in the past is figured out the level of difficulty they think they can handle, based on previous efforts, and then strategize to only answer the questions that approach or meet that effort, hoping for a good score because of their work.

For example, on the last test there were 10 questions with the following point and difficulty levels:

- Q#1: 12 points, difficulty level 8
- Q#2: 10 points, difficulty level 5
- Q#3: 8 points, difficulty level 3
- Q#4: 12 points, difficulty level 4
- Q#5: 7 points, difficulty level 5
- Q#6: 13 points, difficulty level 3
- Q#7: 16 points, difficulty level 2
- Q#8: 2 points, difficulty level 8
- Q#9: 14 points, difficulty level 4
- Q#10: 4 points, difficulty level 5

Being very conscientious about their grades, Candela and her friends decide to figure out the best approach to maximize their efforts in getting a decent grade without exceeding their target difficulty level. The teacher will allow students to bring with them the information provided, along with any strategies for which questions to attempt to maximize their test score.

For the last test, the result for Candela was a score of 96, which she was able to achieve by answering all but question #8, for a total of 96 points and an accumulated difficulty level of 39. She will shoot for a 40 level of difficulty for the text next week. Carla, on the other hand had earned a test score of 85 with a combined difficulty level of 29, and therefore thinks 30 is a reasonable difficulty number for her to attempt.

Input and output descriptions and samples on next page.

(Candela – cont.)

Input: The data file will contain an initial integer Q (10 <= Q <= 30), indicating Q questions to follow. Each question data set consists of two values, an integer P representing the number of points that question is worth, an integer D indicating the difficulty level of the question. The first question is Question #1, the second is Question #2, continuing in that sequence, with the last one as Question #Q. Following the list of questions, several integers T will follow, each on one line and each representing a target difficulty level indicated by a student. There will be only one set of questions to process, but several target values after the question listing.

Output: For each data set, list the target difficulty designated, the calculated difficulty expected, the total score of the questions the student has selected to attempt and hopes to achieve, and each question on the calculated list, shown with the points and difficulty level, exactly as displayed, formatted and aligned in the examples below. Print a final “**=====**” line below each complete output.

Sample input:

```
10
12 8
10 5
8 3
12 4
7 5
13 3
16 2
2 8
14 4
4 5
40
30
10
```

Sample output:

```
Target diff     = 40
Calculated diff = 39
Expected points = 96
Q# 1, 12 pts, diff 8
Q# 2, 10 pts, diff 5
Q# 3,  8 pts, diff 3
Q# 4, 12 pts, diff 4
Q# 5,  7 pts, diff 5
Q# 6, 13 pts, diff 3
Q# 7, 16 pts, diff 2
Q# 9, 14 pts, diff 4
Q#10,  4 pts, diff 5
=====
Target diff     = 30
Calculated diff = 29
Expected points = 85
Q# 1, 12 pts, diff 8
Q# 2, 10 pts, diff 5
Q# 3,  8 pts, diff 3
Q# 4, 12 pts, diff 4
Q# 6, 13 pts, diff 3
Q# 7, 16 pts, diff 2
Q# 9, 14 pts, diff 4
=====
Target diff     = 10
Calculated diff = 9
Expected points = 43
Q# 6, 13 pts, diff 3
Q# 7, 16 pts, diff 2
Q# 9, 14 pts, diff 4
=====
```

4. Carla

Program Name: Carla.java Input File: carla.dat

In the UNIX operating system, Carla has recently learned, each file, directory, or link is "owned" by a "user", who is a member of a "group", and has certain "permissions" assigned to it, represented by a 10-character string, such as "drwxrwxrwx". The first character is 'd', '-', or 'l' (directory, file, or link), followed by three sets of "rwx" values, indicating "read, write, execute" permissions. The first set is the user's rights, the middle set the group's rights, and the third everyone else's rights to that object.

Permission denied for any of these rights is represented by a '-'in place of the 'r', 'w', or 'x'. For example, a sample directory permission string would be "drwxr--r--", indicating full directory rights for the user, but "read-only" rights for the group member and all others.

Each "rwx" combination can also be represented by an octal value (0-7), as shown below:

Octal value	r w x combination	Interpretation
0	- - -	No permission
1	- - 1	Execute permission only
2	- 1 -	Write permission only
3	- 1 1	Write and execute permission
4	1 - -	Read-only permission
5	1 - 1	Read and execute only
6	1 1 -	Read and write only
7	1 1 1	Full permission

Given a four-character code string made up of a character 'D, 'F' or 'L', followed by a 3-digit octal integer value, like 664, output the resulting 10 character string that represents the permission value indicated.

Input: Several four-character codes as described above, each on one line.

Output: The resulting 10-character string based on the criteria described above.

Sample input:

```
F664
D775
L334
F530
D127
```

Sample output:

```
-rw-rw-r--
drwxrwxr-x
l-wx-wxr--
-r-x-wx---
d--x-w-rwx
```

5. Diya

Program Name: Diya.java Input File: diya.dat

Diya has decided to take on the challenge of producing the classic spiral matrix, a series of consecutive integers starting with 1 at the top left of the square, going across the top and down the right side, around and around until the square of the side length of the square is in the very center. He needs your help to write this program. You up to the challenge?

Input: Several integers N, each on a separate line, 2 <= N < 20.

Output: For each integer, output a spiral matrix as indicated above, and shown below. All column values must be left justified, with exactly one space following the largest value in the center of the output and all other columns consistently spaced with the column containing this value. Print a final "**=====**" line below each complete output.

Sample input:

```
3
6
10
```

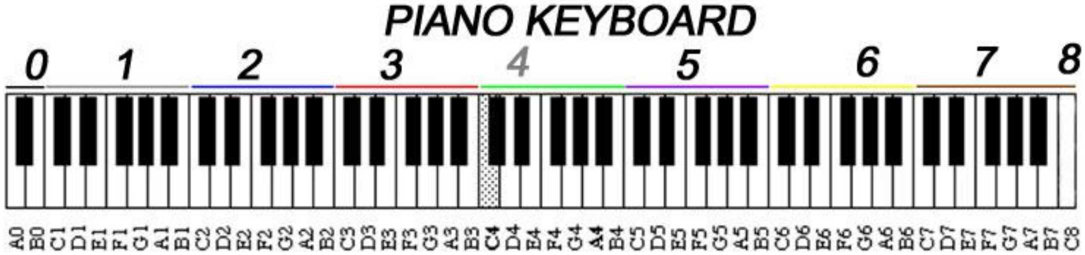
Sample output:

```
1 2 3
8 9 4
7 6 5
=====
1  2  3  4  5  6
20 21 22 23 24 7
19 32 33 34 25 8
18 31 36 35 26 9
17 30 29 28 27 10
16 15 14 13 12 11
=====
1   2   3   4   5   6   7   8   9   10
36  37  38  39  40  41  42  43  44  11
35  64  65  66  67  68  69  70  45  12
34  63  84  85  86  87  88  71  46  13
33  62  83  96  97  98  89  72  47  14
32  61  82  95  100 99  90  73  48  15
31  60  81  94  93  92  91  74  49  16
30  59  80  79  78  77  76  75  50  17
29  58  57  56  55  54  53  52  51  18
28  27  26  25  24  23  22  21  20  19
=====
```

6. Gleb

Program Name: Gleb.java Input File: gleb.dat

Having studied piano for several years, Gleb knows that the white keys on the standard 88-key piano are arranged in diatonic scales from C to B (C,D,E,F,G,A,B), each group of seven letters numbered from 0 to 8, starting with group zero with only two notes, A0 as the very lowest key, then B0, followed by group one with C1 through B1, group two, C2 to B2, C3 to B3, all the way to C8 as the highest key. A "C" scale starting at C3 would be C3, D3, E3, F3, G3, A3, B3, and C4 (which is middle C on the piano).


PIANO KEYBOARD
0
1
2
3
4
5
6
7
8
A0 B0 C1 D1 E1 F1 G1 A1 B1 C2 D2 E2 F2 G2 A2 B2 C3 D3 E3 F3 G3 A3 B3 C4 D4 E4 F4 G4 A4 B4 C5 D5 E5 F5 G5 A5 B5 C6 D6 E6 F6 G6 A6 B6 C7 D7 E7 F7 G7 A7 B7 C8

For data in a programming project he is planning, he decides to represent a melody with the starting note, followed by several positive or negative integers representing the intervals that follow that note, like 2 for the interval of "up a second", 3 for "up a third", -4 for "down a fourth", etc. The interval of "up a second" simply goes from one note to the next, like C to D, D to E, or A to B. "Up a third" would skip a letter, like going from C to E, or F to A. The rest of the interval jumps progress in the same way. *(There are further interval designations like "major", "minor", "perfect", "augmented" and "diminished", but for now we'll just stick to the basic white key intervals. Also, rhythms will not be included at this point in the project.)*

Gleb's goal is to create an alphanumeric text stream that could be interpreted by a melody function that would sound the actual notes. For example, the data for the melody, "The Eyes of Texas", would begin like this, with the actual notes produced shown below the words:

```
C4  4    -4 4   -4 4   2  2  -3
The eyes of Tex-as are up-on you
C4  F4   C4 F4  C4 F4  G4 A4 F4
```

C4 is the starting note, followed by 4 which indicates an interval of "up a fourth" to F4, -4 going back down to C4, and so on. The output for the above input would be:

```
C4 F4 C4 F4 C4 F4 G4 A4 F4
```

Input: Several lines of data, each line containing a beginning "note" (letter, integer), followed by several positive or negative integers, no more than 25 notes in the entire melody.

Output: A stream of letter/number combinations representing the actual melody represented by the data. All letters must be uppercase, and at least one space must separate each "note".

Sample input:

```
C4 4 -4 4 -4 4 2 2 -3 (Melody for "The Eyes of Texas")
F5 4 2 -5 4 2 2 -3 2 2 -3 2 (Melody for "Maria", West Side Story)
C6 1 1 -4 2 1 -2 6 1 -2 1 -2 (Melody for "Old MacDonald Had A Farm")
```

Sample output:

```
C4 F4 C4 F4 C4 F4 G4 A4 F4
F5 B5 C6 F5 B5 C6 D6 B5 C6 D6 B5 C6
C6 C6 C6 G5 A5 A5 G5 E6 E6 D6 D6 C6
```

7. Jeremy

Program Name: Jeremy.java Input File: jeremy.dat

Jeremy knows that bitmap images can be represented by a matrix of integers, with each integer in the matrix representing the color of a “pixel”. Various editing operations can be performed on a bitmap, but one of the most common ones is the flood fill. He knows that a flood fill is a change process that starts at a single pixel, changing every adjacent pixel that is the same color as the starting pixel *(not including adjacent cells in diagonal directions)*, to another color. The process continues for all the pixels that were changed, until an entire block of color in the picture has been changed. Jeremy needs your help in creating a program that, given the length and width of a bitmap, a matrix of integers from 0-9 that represents the bitmap, the starting pixel, and a “color” from 0-9 to change to, will perform a flood fill operation on that bitmap.

For example, in this 4 x 7 matrix, the color at position (2,2) is a 4, as shown in bold. If this location was changed to the color 6, and then a flood fill is performed for all neighboring values of 4, resulting in the second matrix. The remaining 4 in the top right corner stays unchanged since it is not reachable.

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

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

Input: An initial integer N, representing N data sets to follow. Each data set consists of two integers R and C, indicating an R x C matrix of integers, which follows on the next R rows. The matrix consists of single integers in the range 0-9, with single space separation. Following the matrix are two integers A and B representing the target location in the matrix, followed by an integer D as the flood fill color.

Output: The resulting matrix after the flood fill operation, which changes all instances of the color integer at location (A,B) to the color integer D, as described above and shown in the examples below. Print a final “=====” line below each complete output.

Sample input:

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

Sample output:

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

8. Kinga

Program Name: Kinga.java Input File: kinga.dat

Kinga knows that boolean variables are those that are either true or false, often represented as 1 or 0. In combination, two variables can have four possibilities, as shown in the table below.

```
A|B
0|0
0|1
1|0
1|1
```

She decides to write a program to output all possible combinations for up to nine variables, in table format, starting with all zeroes on the top row, and all 1s on the bottom row. She decides on the output format shown below, with A, B, C, and so on representing the variables, 0 for false, 1 for true, and the correct sequence of combinations in logical order. Columns are separated by the "|" symbol. Her input data will consist of one integer, between 1 and 9, inclusive, representing the number of boolean variables.

Input: Several integers N, 1<=N<=9.

Output: The corresponding Boolean truth table representing all possible true/false combinations, as described above and shown in the examples below. Print a final "**=====**" line below each complete output.

Sample input:

```
3
4
```

Sample output:

```
  A|B|C
1 0|0|0
2 0|0|1
3 0|1|0
4 0|1|1
5 1|0|0
6 1|0|1
7 1|1|0
8 1|1|1
=====
```

```
  A|B|C|D
1 0|0|0|0
2 0|0|0|1
3 0|0|1|0
4 0|0|1|1
5 0|1|0|0
6 0|1|0|1
7 0|1|1|0
8 0|1|1|1
9 1|0|0|0
10 1|0|0|1
11 1|0|1|0
12 1|0|1|1
13 1|1|0|0
14 1|1|0|1
15 1|1|1|0
16 1|1|1|1
=====
```

9. Layla

Program Name: Layla.java   Input File: layla.dat

Layla is considering a thought experiment in measurement systems, different than metric or the traditional English system, perhaps ones that remote civilizations, or even aliens on different worlds might develop. She wants to allow for three levels of measure, like meters, kilometers, and centimeters, all members of the metric system. Just as it takes 100 centimeters to make a meter, and 1000 meters to make a kilometer, she wants to consider other systems with other conversion values.

For purposes of consistency, she decides to label the three units of measure as A, B and C, and then express the conversions as follows: B = xA and C = yB, where x and y are real values greater than 1.

In the metric system, A, B and C would be centimeters, meters and kilometers, and x and y would be 100 and 1000. To express 5 meters in terms of centimeters and kilometers, she would mathematically convert them using the values of 100 and 1000, resulting in 500 A units (centimeters), and 0.005 C units (kilometers). In her experiment, A will always be the smallest unit and C the largest.

In one possible random system, the values of x and y might be 16 and 7, which would mean that B = 16A, and C = 7B. She then would take a value, express it in one measure, and then convert it into equivalent values in the other two measures. 17 units of B would convert into 272 A units and 2.429 C units. Three C units in the same system would be equivalent to 336 A units and 21 B units.

Input: Several sets of data, each on one line, consisting of two integer values **x** and **y**, a value **d**, and a character **c**, all with single space separation.

Output: The equivalent values in all three units, A, B and C, for the given data set, rounded to three places of precision, and output as shown below. Print a final "**=====**" line below each complete output.

Sample input:

```
100 1000 5 B
16 7 17 B
16 7 3 C
10 6 4.25 A
```

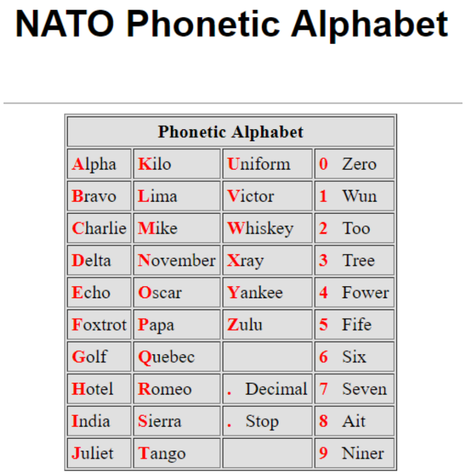
Sample output:

```
A = 500.000
B = 5.000
C = 0.005
=====
A = 272.000
B = 17.000
C = 2.429
=====
A = 336.000
B = 21.000
C = 3.000
=====
A = 4.250
B = 0.425
C = 0.071
=====
```

10. Max

Program Name: Max.java Input File: max.dat

In ROTC class, Max has been learning how the military and other organizations use special words to represent letters of the English alphabet and the digits of the base ten number system. A special word corresponds to each symbol, each unique in its sound, created to better ensure reliable radio communication, especially when crucial information is being transmitted and received over systems that encounter noise and interference, like pilots talking to control towers, military personnel calling in air strike or rescue locations, police communicating a license plate number over the radio, ship captains communicating with other vessels while traversing the local waterways, or someone giving a credit card number over the phone for an important purchase. Over the years, many different systems have been developed, but the system shown here is the latest one adopted by NATO and used worldwide by many.



NATO Phonetic Alphabet

Phonetic Alphabet			
Alpha	Kilo	Uniform	0 Zero
Bravo	Lima	Victor	1 Wun
Charlie	Mike	Whiskey	2 Too
Delta	November	Xray	3 Tree
Echo	Oscar	Yankee	4 Fower
Foxtrot	Papa	Zulu	5 Fife
Golf	Quebec		6 Six
Hotel	Romeo	. Decimal	7 Seven
India	Sierra	. Stop	8 Ait
Juliet	Tango		9 Niner

Max has a verbal test coming up and needs to practice communicating various information using these words. He wants to write a program to input an alphanumeric string and produce the correct series of NATO phonetic words to communicate the message. Can you help?

Input: Several single alphanumeric strings, each on one line.

Output: The corresponding series of phonetic words that represent the string, with single space separation.

Sample input:

```
ABC
DBD7555
54331234
TX1041HU
```

Sample output:

```
Alpha Bravo Charlie
Delta Bravo Delta Seven Fife Fife Fife
Fife Fower Tree Tree Wun Too Tree Fower
Tango Xray Wun Zero Fower Wun Hotel Uniform
```

11. Nandita

Program Name: Nandita.java **Input File: nandita.dat**

Nandita has learned that in some areas of the world the standard format for abbreviating a date differs from others, like the traditional month/day/year abbreviation method used often in the US. For example, in her research she has discovered that some may express **APRIL 15, 2018** as **04/15/18** (called “middle endian” format), others may use **15.04.2018** (“little endian” format), and still others **2018-04-15** (“big endian”).

middle-endian (month, day, year), *e.g. 04/15/18*
little-endian (day, month, year), *e.g. 15.04.2018*
big-endian (year, month, day), *e.g. 2018-04-15*

Given a day of the year expressed fully, such as **`APRIL 15, 2018`**, show it in each of the abbreviated formats described above, in the order middle endian, little endian, big endian.

Input: Several dates fully expressed, as described above and shown in the examples below. All month names will be uppercased, fully spelled out, followed by one space, the day number, a comma and space, then the four-digit year number. Each input data set is all on one line.

Output: The given date abbreviated in three different formats: middle endian, little endian and big endian. Print a final “**=====**” line below each complete output.

Sample input:

```
APRIL 15, 2018
DECEMBER 7, 1941
SEPTEMBER 11, 2001
```

Sample output:

```
04/15/18
15.04.2018
2018-04-15
=====
12/07/41
07.12.1941
1941-12-07
=====
09/11/01
11.09.2001
2001-09-11
=====
```

12. Raymond

Program Name: Raymond.java Input File: raymond.dat

Raymond has just learned about complement values, with 12 and -13 being complements of each other, -46 the complement of 45, 8 the complement of -9, and so on. He wants to write program to output an integer value and its complement, but needs your help. Please?

Input: Several integers N, all on one line, with single space separation.

Output: The input value N, followed by a single space, followed by its complement value.

Sample input:

```
12 45 -9
```

Sample output:

```
12 -13
45 -46
-9 8
```

UIL Computer Science Competition

District 2018

JUDGES PACKET - CONFIDENTIAL

I. Instructions

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

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

II. Table of Contents

Problem #1
60 Points

1. Alice

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

Test Output To Screen

```
           &
           &&
           &&&
           &&-&
           &&--&
           &&---&
           &&----&
           &&--.--&
           &&--..--&
           &&--...--&
           &&--....--&
           &&--.....--&
           &&--......--&
           &&--.......--&
           &&--........--&
           &&--.........--&
           &&--..........--&
           &&--...........--&
           &&--............--&
           &&--.............--&
            ||                    \o/
            ||                     |
 ======================================
   ==================================
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

Problem #2
60 Points

2. Bayani

Program Name: Bayani.java **Input File: bayani.dat**

Test Input File:

```
6.99
12.87
5.44
99.99
115.87
564.00
348.24
5.13
0.78
90.54
32.10
77.79
```

Test Output To Screen

```
        $   6.99
        $  12.87
        $   5.44
        $  99.99
        $ 115.87
        $ 564.00
        $ 348.24
        $   5.13
        $   0.78
        $  90.54
        $  32.10
        $  77.79
Total = $1359.74
```

Problem #3
60 Points

3. Candela

Program Name: Candela.java **Input File: candela.dat**

Test Input File:

```
10
12 8
10 5
8 3
12 4
7 5
13 3
16 2
2 8
14 4
4 5
40
30
10
50
25
26
29
14
13
12
```

Test Output To Screen

```
Target diff     = 40
Calculated diff = 39
Expected points = 96
Q# 1, 12 pts, diff 8
Q# 2, 10 pts, diff 5
Q# 3,  8 pts, diff 3
Q# 4, 12 pts, diff 4
Q# 5,  7 pts, diff 5
Q# 6, 13 pts, diff 3
Q# 7, 16 pts, diff 2
Q# 9, 14 pts, diff 4
Q#10,  4 pts, diff 5
=====
Target diff     = 30
Calculated diff = 29
Expected points = 85
Q# 1, 12 pts, diff 8
Q# 2, 10 pts, diff 5
Q# 3,  8 pts, diff 3
Q# 4, 12 pts, diff 4
Q# 6, 13 pts, diff 3
Q# 7, 16 pts, diff 2
Q# 9, 14 pts, diff 4
=====
Target diff     = 10
Calculated diff = 9
Expected points = 43
Q# 6, 13 pts, diff 3
Q# 7, 16 pts, diff 2
Q# 9, 14 pts, diff 4
=====
Target diff     = 50
Calculated diff = 47
Expected points = 98
Q# 1, 12 pts, diff 8
Q# 2, 10 pts, diff 5
Q# 3,  8 pts, diff 3
Q# 4, 12 pts, diff 4
Q# 5,  7 pts, diff 5
Q# 6, 13 pts, diff 3
Q# 7, 16 pts, diff 2
Q# 8,  2 pts, diff 8
Q# 9, 14 pts, diff 4
Q#10,  4 pts, diff 5
=====
Target diff     = 25
Calculated diff = 24
Expected points = 75
Q# 1, 12 pts, diff 8
Q# 3,  8 pts, diff 3
Q# 4, 12 pts, diff 4
Q# 6, 13 pts, diff 3
Q# 7, 16 pts, diff 2
Q# 9, 14 pts, diff 4
=====
Target diff     = 26
Calculated diff = 26
Expected points = 80
Q# 2, 10 pts, diff 5
Q# 3,  8 pts, diff 3
Q# 4, 12 pts, diff 4
Q# 5,  7 pts, diff 5
Q# 6, 13 pts, diff 3
Q# 7, 16 pts, diff 2
Q# 9, 14 pts, diff 4
=====
Target diff     = 29
Calculated diff = 29
Expected points = 85
Q# 1, 12 pts, diff 8
Q# 2, 10 pts, diff 5
Q# 3,  8 pts, diff 3
Q# 4, 12 pts, diff 4
Q# 6, 13 pts, diff 3
Q# 7, 16 pts, diff 2
Q# 9, 14 pts, diff 4
=====
Target diff     = 14
Calculated diff = 14
Expected points = 50
Q# 5,  7 pts, diff 5
Q# 6, 13 pts, diff 3
Q# 7, 16 pts, diff 2
Q# 9, 14 pts, diff 4
=====
Target diff     = 13
Calculated diff = 13
Expected points = 55
Q# 4, 12 pts, diff 4
Q# 6, 13 pts, diff 3
Q# 7, 16 pts, diff 2
Q# 9, 14 pts, diff 4
=====
Target diff     = 12
Calculated diff = 12
Expected points = 16
Q# 8,  2 pts, diff 8
Q# 9, 14 pts, diff 4
=====
```

Problem #4
60 Points

4. Carla

Program Name: Carla.java **Input File: carla.dat**

Test Input File:

```
F664
D775
L334
F530
D127
F100
D010
L001
F777
L036
```

Test Output To Screen

```
-rw-rw-r--
drwxrwxr-x
l-wx-wxr--
-r-x-wx---
d--x-w-rwx
---x------
d-----x---
l--------x
-rwxrwxrwx
l----wxrw-
```

Problem #5
60 Points

5. Diya

Program Name: Diya.java **Input File: diya.dat**

Test Input File:

```
3
6
10
9
15
19
```

Test Output To Screen

```
1 2 3
8 9 4
7 6 5
=====
1  2  3  4  5  6
20 21 22 23 24 7
19 32 33 34 25 8
18 31 36 35 26 9
17 30 29 28 27 10
16 15 14 13 12 11
=====
1   2   3   4   5   6   7   8   9   10
36  37  38  39  40  41  42  43  44  11
35  64  65  66  67  68  69  70  45  12
34  63  84  85  86  87  88  71  46  13
33  62  83  96  97  98  89  72  47  14
32  61  82  95  100 99  90  73  48  15
31  60  81  94  93  92  91  74  49  16
30  59  80  79  78  77  76  75  50  17
29  58  57  56  55  54  53  52  51  18
28  27  26  25  24  23  22  21  20  19
=====
1  2  3  4  5  6  7  8  9
32 33 34 35 36 37 38 39 10
31 56 57 58 59 60 61 40 11
30 55 72 73 74 75 62 41 12
29 54 71 80 81 76 63 42 13
28 53 70 79 78 77 64 43 14
27 52 69 68 67 66 65 44 15
26 51 50 49 48 47 46 45 16
25 24 23 22 21 20 19 18 17
=====
1   2   3   4   5   6   7   8   9   10  11  12  13  14  15
56  57  58  59  60  61  62  63  64  65  66  67  68  69  16
55  104 105 106 107 108 109 110 111 112 113 114 115 70  17
54  103 144 145 146 147 148 149 150 151 152 153 116 71  18
53  102 143 176 177 178 179 180 181 182 183 154 117 72  19
52  101 142 175 200 201 202 203 204 205 184 155 118 73  20
51  100 141 174 199 216 217 218 219 206 185 156 119 74  21
50  99  140 173 198 215 224 225 220 207 186 157 120 75  22
```

```
49  98  139 172 197 214 223 222 221 208 187 158 121 76  23
48  97  138 171 196 213 212 211 210 209 188 159 122 77  24
47  96  137 170 195 194 193 192 191 190 189 160 123 78  25
46  95  136 169 168 167 166 165 164 163 162 161 124 79  26
45  94  135 134 133 132 131 130 129 128 127 126 125 80  27
44  93  92  91  90  89  88  87  86  85  84  83  82  81  28
43  42  41  40  39  38  37  36  35  34  33  32  31  30  29
=====
1   2   3   4   5   6   7   8   9   10  11  12  13  14  15  16  17  18  19
72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  20
71  136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 90  21
70  135 192 193 194 195 196 197 198 199 200 201 202 203 204 205 152 91  22
69  134 191 240 241 242 243 244 245 246 247 248 249 250 251 206 153 92  23
68  133 190 239 280 281 282 283 284 285 286 287 288 289 252 207 154 93  24
67  132 189 238 279 312 313 314 315 316 317 318 319 290 253 208 155 94  25
66  131 188 237 278 311 336 337 338 339 340 341 320 291 254 209 156 95  26
65  130 187 236 277 310 335 352 353 354 355 342 321 292 255 210 157 96  27
64  129 186 235 276 309 334 351 360 361 356 343 322 293 256 211 158 97  28
63  128 185 234 275 308 333 350 359 358 357 344 323 294 257 212 159 98  29
62  127 184 233 274 307 332 349 348 347 346 345 324 295 258 213 160 99  30
61  126 183 232 273 306 331 330 329 328 327 326 325 296 259 214 161 100 31
60  125 182 231 272 305 304 303 302 301 300 299 298 297 260 215 162 101 32
59  124 181 230 271 270 269 268 267 266 265 264 263 262 261 216 163 102 33
58  123 180 229 228 227 226 225 224 223 222 221 220 219 218 217 164 103 34
57  122 179 178 177 176 175 174 173 172 171 170 169 168 167 166 165 104 35
56  121 120 119 118 117 116 115 114 113 112 111 110 109 108 107 106 105 36
55  54  53  52  51  50  49  48  47  46  45  44  43  42  41  40  39  38  37
=====
```

Problem #6
60 Points

6. Gleb

Program Name: Gleb.java Input File: gleb.dat

Test Input File:

C4 4 –4 4 –4 4 2 2 –3 *(Melody for "The Eyes of Texas")*
F5 4 2 –5 4 2 2 –3 2 2 –3 2 *("Maria", West Side Story)*
C6 1 1 –4 2 1 –2 6 1 –2 1 –2 *("Old MacDonald Had A Farm")*
E4 -2 2 2 -2 -2 2 -3 2 1 2 *("Finlandia")*
G3 6 -2 -2 2 -2 -3 -2 -3 3 6 -2 -2 1 -2 2 2 *("My Bonnie Lies Over The Ocean")*
C5 8 -2 -3 2 2 2 -8 6 -2 -7 6 -2 -3 2 2 2 -3 -3 2 2 2 -3*("Somewhere Over The Rainbow")*

Test Output To Screen

```
C4 F4 C4 F4 C4 F4 G4 A4 F4
F5 B5 C6 F5 B5 C6 D6 B5 C6 D6 B5 C6
C6 C6 C6 G5 A5 A5 G5 E6 E6 D6 D6 C6
E4 D4 E4 F4 E4 D4 E4 C4 D4 D4 E4
G3 E4 D4 C4 D4 C4 A3 G3 E3 G3 E4 D4 C4 C4 B3 C4 D4
C5 C6 B5 G5 A5 B5 C6 C5 A5 G5 A4 F5 E5 C5 D5 E5 F5 D5 B4 C5 D5 E5 C5
```

Problem #7
60 Points

7. Jeremy

Program Name: Jeremy.java **Input File: jeremy.dat**

Test Input File:

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

Test Output To Screen

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

Problem #8
60 Points

8. Kinga

Program Name: Kinga.java Input File: kinga.dat

Test Input File:

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

Test Output To Screen

```
  A|B|C
1 0|0|0
2 0|0|1
3 0|1|0
4 0|1|1
5 1|0|0
6 1|0|1
7 1|1|0
8 1|1|1
=====
  A|B|C|D
1 0|0|0|0
2 0|0|0|1
3 0|0|1|0
4 0|0|1|1
5 0|1|0|0
6 0|1|0|1
7 0|1|1|0
8 0|1|1|1
9 1|0|0|0
10 1|0|0|1
11 1|0|1|0
12 1|0|1|1
13 1|1|0|0
14 1|1|0|1
15 1|1|1|0
16 1|1|1|1
=====
  A|B|C|D|E
1 0|0|0|0|0
2 0|0|0|0|1
3 0|0|0|1|0
4 0|0|0|1|1
5 0|0|1|0|0
6 0|0|1|0|1
7 0|0|1|1|0
8 0|0|1|1|1
9 0|1|0|0|0
10 0|1|0|0|1
11 0|1|0|1|0
12 0|1|0|1|1
13 0|1|1|0|0
14 0|1|1|0|1
15 0|1|1|1|0
16 0|1|1|1|1
17 1|0|0|0|0
18 1|0|0|0|1
19 1|0|0|1|0
20 1|0|0|1|1
21 1|0|1|0|0
22 1|0|1|0|1
23 1|0|1|1|0
24 1|0|1|1|1
25 1|1|0|0|0
26 1|1|0|0|1
27 1|1|0|1|0
28 1|1|0|1|1
29 1|1|1|0|0
30 1|1|1|0|1
31 1|1|1|1|0
32 1|1|1|1|1
=====
  A|B|C|D|E|F
1 0|0|0|0|0|0
2 0|0|0|0|0|1
3 0|0|0|0|1|0
4 0|0|0|0|1|1
5 0|0|0|1|0|0
6 0|0|0|1|0|1
7 0|0|0|1|1|0
8 0|0|0|1|1|1
9 0|0|1|0|0|0
10 0|0|1|0|0|1
11 0|0|1|0|1|0
12 0|0|1|0|1|1
13 0|0|1|1|0|0
14 0|0|1|1|0|1
15 0|0|1|1|1|0
16 0|0|1|1|1|1
17 0|1|0|0|0|0
18 0|1|0|0|0|1
19 0|1|0|0|1|0
20 0|1|0|0|1|1
21 0|1|0|1|0|0
22 0|1|0|1|0|1
23 0|1|0|1|1|0
24 0|1|0|1|1|1
25 0|1|1|0|0|0
26 0|1|1|0|0|1
27 0|1|1|0|1|0
28 0|1|1|0|1|1
29 0|1|1|1|0|0
30 0|1|1|1|0|1
31 0|1|1|1|1|0
32 0|1|1|1|1|1
33 1|0|0|0|0|0
34 1|0|0|0|0|1
35 1|0|0|0|1|0
36 1|0|0|0|1|1
37 1|0|0|1|0|0
38 1|0|0|1|0|1
39 1|0|0|1|1|0
40 1|0|0|1|1|1
41 1|0|1|0|0|0
42 1|0|1|0|0|1
43 1|0|1|0|1|0
44 1|0|1|0|1|1
45 1|0|1|1|0|0
46 1|0|1|1|0|1
47 1|0|1|1|1|0
48 1|0|1|1|1|1
49 1|1|0|0|0|0
50 1|1|0|0|0|1
51 1|1|0|0|1|0
52 1|1|0|0|1|1
53 1|1|0|1|0|0
54 1|1|0|1|0|1
55 1|1|0|1|1|0
56 1|1|0|1|1|1
57 1|1|1|0|0|0
58 1|1|1|0|0|1
59 1|1|1|0|1|0
60 1|1|1|0|1|1
61 1|1|1|1|0|0
62 1|1|1|1|0|1
63 1|1|1|1|1|0
64 1|1|1|1|1|1
=====
  A|B|C|D|E|F|G
1 0|0|0|0|0|0|0
2 0|0|0|0|0|0|1
3 0|0|0|0|0|1|0
4 0|0|0|0|0|1|1
5 0|0|0|0|1|0|0
6 0|0|0|0|1|0|1
7 0|0|0|0|1|1|0
8 0|0|0|0|1|1|1
```

```
9 0|0|0|1|0|0|0
10 0|0|0|1|0|0|1
11 0|0|0|1|0|1|0
12 0|0|0|1|0|1|1
13 0|0|0|1|1|0|0
14 0|0|0|1|1|0|1
15 0|0|0|1|1|1|0
16 0|0|0|1|1|1|1
17 0|0|1|0|0|0|0
18 0|0|1|0|0|0|1
19 0|0|1|0|0|1|0
20 0|0|1|0|0|1|1
21 0|0|1|0|1|0|0
22 0|0|1|0|1|0|1
23 0|0|1|0|1|1|0
24 0|0|1|0|1|1|1
25 0|0|1|1|0|0|0
26 0|0|1|1|0|0|1
27 0|0|1|1|0|1|0
28 0|0|1|1|0|1|1
29 0|0|1|1|1|0|0
30 0|0|1|1|1|0|1
31 0|0|1|1|1|1|0
32 0|0|1|1|1|1|1
33 0|1|0|0|0|0|0
34 0|1|0|0|0|0|1
35 0|1|0|0|0|1|0
36 0|1|0|0|0|1|1
37 0|1|0|0|1|0|0
38 0|1|0|0|1|0|1
39 0|1|0|0|1|1|0
40 0|1|0|0|1|1|1
41 0|1|0|1|0|0|0
42 0|1|0|1|0|0|1
43 0|1|0|1|0|1|0
44 0|1|0|1|0|1|1
45 0|1|0|1|1|0|0
46 0|1|0|1|1|0|1
47 0|1|0|1|1|1|0
48 0|1|0|1|1|1|1
49 0|1|1|0|0|0|0
50 0|1|1|0|0|0|1
51 0|1|1|0|0|1|0
52 0|1|1|0|0|1|1
53 0|1|1|0|1|0|0
54 0|1|1|0|1|0|1
55 0|1|1|0|1|1|0
56 0|1|1|0|1|1|1
57 0|1|1|1|0|0|0
58 0|1|1|1|0|0|1
59 0|1|1|1|0|1|0
60 0|1|1|1|0|1|1
61 0|1|1|1|1|0|0
62 0|1|1|1|1|0|1
63 0|1|1|1|1|1|0
64 0|1|1|1|1|1|1
65 1|0|0|0|0|0|0
66 1|0|0|0|0|0|1
67 1|0|0|0|0|1|0
68 1|0|0|0|0|1|1
69 1|0|0|0|1|0|0
70 1|0|0|0|1|0|1
71 1|0|0|0|1|1|0
72 1|0|0|0|1|1|1
73 1|0|0|1|0|0|0
74 1|0|0|1|0|0|1
75 1|0|0|1|0|1|0
76 1|0|0|1|0|1|1
77 1|0|0|1|1|0|0
78 1|0|0|1|1|0|1
79 1|0|0|1|1|1|0
80 1|0|0|1|1|1|1
81 1|0|1|0|0|0|0
82 1|0|1|0|0|0|1
83 1|0|1|0|0|1|0
84 1|0|1|0|0|1|1
85 1|0|1|0|1|0|0
86 1|0|1|0|1|0|1
87 1|0|1|0|1|1|0
88 1|0|1|0|1|1|1
89 1|0|1|1|0|0|0
90 1|0|1|1|0|0|1
91 1|0|1|1|0|1|0
92 1|0|1|1|0|1|1
93 1|0|1|1|1|0|0
94 1|0|1|1|1|0|1
95 1|0|1|1|1|1|0
96 1|0|1|1|1|1|1
97 1|1|0|0|0|0|0
98 1|1|0|0|0|0|1
99 1|1|0|0|0|1|0
100 1|1|0|0|0|1|1
101 1|1|0|0|1|0|0
102 1|1|0|0|1|0|1
103 1|1|0|0|1|1|0
104 1|1|0|0|1|1|1
105 1|1|0|1|0|0|0
106 1|1|0|1|0|0|1
107 1|1|0|1|0|1|0
108 1|1|0|1|0|1|1
109 1|1|0|1|1|0|0
110 1|1|0|1|1|0|1
111 1|1|0|1|1|1|0
112 1|1|0|1|1|1|1
113 1|1|1|0|0|0|0
114 1|1|1|0|0|0|1
115 1|1|1|0|0|1|0
116 1|1|1|0|0|1|1
117 1|1|1|0|1|0|0
118 1|1|1|0|1|0|1
119 1|1|1|0|1|1|0
120 1|1|1|0|1|1|1
121 1|1|1|1|0|0|0
122 1|1|1|1|0|0|1
123 1|1|1|1|0|1|0
124 1|1|1|1|0|1|1
125 1|1|1|1|1|0|0
126 1|1|1|1|1|0|1
127 1|1|1|1|1|1|0
128 1|1|1|1|1|1|1
=====
  A|B
1 0|0
2 0|1
3 1|0
4 1|1
=====
  A
1 0
2 1
=====
```

Problem #9
60 Points

9. Layla

Program Name: Layla.java   Input File: layla.dat

Test Input File:

```
100 1000 5 B
16 7 17 B
16 7 3 C
10 6 4.25 A
10 8 300 A
14 3 100 A
22 12 100 B
24 5 100 C
9 9 999 A
8 7 99.9 B
5 6 9.99 C
4 3 0.987 B
```

Test Output To Screen

```
A = 500.000
B = 5.000
C = 0.005
=====
A = 272.000
B = 17.000
C = 2.429
=====
A = 336.000
B = 21.000
C = 3.000
=====
A = 4.250
B = 0.425
C = 0.071
=====
A = 300.000
B = 30.000
C = 3.750
=====
A = 100.000
B = 7.143
C = 2.381
=====
A = 2200.000
B = 100.000
C = 8.333
=====
A = 12000.000
B = 500.000
C = 100.000
=====
A = 999.000
B = 111.000
C = 12.333
=====
A = 799.200
B = 99.900
C = 14.271
=====
A = 299.700
B = 59.940
C = 9.990
=====
A = 3.948
B = 0.987
C = 0.329
=====
```

Problem #10
60 Points

10. Max

Program Name: Max.java **Input File: max.dat**

Test Input File:

```
ABC
DBD7555
54331234
TX1041HU
ZYWX802
ECHO5
CHARLIE
TUV594FG
JK6MNP7QS
```

Test Output To Screen

```
Alpha Bravo Charlie
Delta Bravo Delta Seven Fife Fife Fife
Fife Fower Tree Tree Wun Too Tree Fower
Tango Xray Wun Zero Fower Wun Hotel Uniform
Zulu Yankee Whiskey Xray Ait Zero Too
Echo Charlie Hotel Oscar Fife
Charlie Hotel Alpha Romeo Lima India Echo
Tango Uniform Victor Fife Niner Fower Foxtrot Golf
Juliet Kilo Six Mike November Papa Seven Quebec Sierra
```

Problem #11
60 Points

11. Nandita

Program Name: Nandita.java **Input File: nandita.dat**

Test Input File:

```
APRIL 15, 2018
DECEMBER 7, 1941
SEPTEMBER 11, 2001
OCTOBER 8, 1956
FEBRUARY 28, 2016
MARCH 1, 2016
```

Test Output To Screen

```
04/15/18
15.04.2018
2018-04-15
=====
12/07/41
07.12.1941
1941-12-07
=====
09/11/01
11.09.2001
2001-09-11
=====
10/08/56
08.10.1956
1956-10-08
=====
02/28/16
28.02.2016
2016-02-28
=====
03/01/16
01.03.2016
2016-03-01
=====
```

Problem #12
60 Points

12. Raymond

Program Name: Raymond.java **Input File: raymond.dat**

Test Input File:

```
12 45 -9 2 -34 6 0 -1
```

Test Output To Screen

```
12 -13
45 -46
-9 8
2 -3
-34 33
6 -7
0 -1
-1 0
```