4. Gan

Program Name: Gan.java **Input File: gan.dat**

Gan is doing some work with circles in his math class, specifically calculating circumference and area, along with surface area and volume of spheres. Using the standard formulas of $2\pi r$, πr^2, $4\pi r^2$, $4/3\pi r^3$, and using the same radius value for all four expressions, his homework is to calculate and express to a precision of one decimal point, the outcomes for all of these calculations, and needs your help to write a program to check his work, showing, in a nice clear, well-aligned format, the results of the calculations. He decides to use the value of PI, precise to 15 places, which is 3.141592653589793.

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

Output: For each integer, output the original value as the radius of a circle, followed by the circumference, area, surface area, and sphere volume using the given radius, formatted exactly as shown below, with dash lines above and below the output, arranged, formatted and aligned exactly as shown below.

Sample input:

```
2 5 3 12
```

Sample output:

```
------------------------------------
 2     12.6     12.6     50.3     33.5
 5     31.4     78.5    314.2    523.6
 3     18.8     28.3    113.1    113.1
12     75.4    452.4   1809.6   7238.2
------------------------------------
```

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
```

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
```

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
```

3. Oksana

Program Name: Oksana.java Input File: None

Oksana loves to play the Connect Dots game, but gets really tired of drawing out the dots on paper. She wants to be able to print them out, but needs your help.

Below is a picture of an 8X8 grid of dots. The dots in each row have a space between, but the rows do not. Write a program to output this exact pattern.

Input: None

Output: An 8X8 grid of dots as shown below, with a single space following each dot, but no blank lines between rows.

Expected Output:

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

7. Tomas

Program Name: Tomas.java **Input File: tomas.dat**

In doing some unusual scientific research lately, Tomas has discovered a peculiar sequence in various naturally occurring situations, where certain measurements of growth produce the consistent pattern 1, 1, 1, 3, 5, 9, 17, and so on. He has recorded the data in stages, where stages 1, 2, and 3 are all the value 1, and then stage 4 begins showing observable growth with the value 3 (sum of the previous three values), stage 5 the value 5 (1+1+3), and so on, following the same arithmetic addition sequence. He wants to be able to predict any particular stage, and needs your help. For example, he wants to know what the value would be at up to stage 50 of this unusual growth pattern.

Input: Several integer values N (1 <= N <= 50), each representing a stage of growth according to the pattern described above, each on one line.

Output: The growth size achieved at stage N in the growth pattern described above.

Sample Input:

```
1
3
5
7
9
```

Sample Output:

```
1
1
5
17
57
```

9. Walter

Program Name: Walter.java **Input File: walter.dat**

Walter works at a mail center and must decide how to mail certain items brought in by customers, according to the dimensions of the item, from small post cards to large packages. Some items don't fit any of the categories and are unmailable. If an item somehow fits into two categories, the larger one is to be used.

Below are the category descriptions that Walter can choose from. All given dimension ranges are inclusive and expressed in inches.

SMALL POST CARD: Length must be between 3.5 and 4.25, width between 3.5 and 6, and thickness between .007 and .016.
LARGE POST CARD: Length between 4.25 and 6, width between 6 and 11.5, thickness between .007 and .016.
SMALL ENVELOPE: Length - 3.5 to 6.125, width - 5 to 11.5, thickness - .016 to .25.
LARGE ENVELOPE: Length - 6.125 to 24, width - 11 to 18, thickness - .25 to .5.
SMALL PACKAGE: Item dimensions must exceed all rules for large envelope and the length plus the distance around the sides of the package other than the length equals 84 inches or less.
LARGE PACKAGE: Length plus distance around the other sides of the package exceeds 84 inches but is no more than 130 inches.

Input: Several data sets, each on one line, consisting of the three values of a mail item: length, width, thickness.

Output: The correct mail classification (in all caps) according to the specifications listed above, or the word UNMAILABLE if the item does not fit within any of the descriptions.

Sample Input:

```
4 4 .009
5 7 .013
5 7 .2
10 12 .4
20 20 40
```

Sample Output:

```
SMALL POST CARD
LARGE POST CARD
SMALL ENVELOPE
LARGE ENVELOPE
UNMAILABLE
```

11. Yash

Program Name: Yash.java Input File: yash.dat

Yash has just learned in computer science class about **Order of Magnitude,** or **Big O,** and wants to work out the numbers. For example, he has learned that any algorithm with an efficiency of O(1) generally takes 1 step to complete, regardless of the size of the data, with varying larger values for the other levels of efficiency, such as O(log N), O(N), O(NlogN), and O(N^2). He learned in class that for a data set of 10 items, these five values are 1, 4, 10, 40, and 100. He was bit confused by O(log N), until his teacher said, *"Think about the exponent for the power of 2 that equals or just exceeds 10."* He thought, *"OK, 2^1 is 2, 2^2 is 4, 2^3 is 8, and 2^4 is 16. That's why 4 is the log base 2, or O(logN), answer for the value 10! I get it! It's just the integer exponent of the number using base 2, that creates a value equal to or just past the number."*

He then tried to work out higher values of N, but started to get confused again, and needs your help.

Input: Several integers N, each on one line, representing the number of elements of data to be processed by an algorithm.

Output: The five values associated with five levels of Big O algorithm efficiency, as described above, for each value N, with values exceeding 999 shown using comma separation, and a single space between each value.

Sample Input:

```
10
50
100
```

Sample Output:

```
1 4 10 40 100
1 6 50 300 2,500
1 7 100 700 10,000
```

2. Alok

Program Name: Alok.java Input File: alok.dat

Alok's classmates are collecting donations to purchase a new computer for a friend whose computer was stolen. He needs a program to calculate the total amount of the donations and wants to know the average amount collected per classmate.

Input: A list of several donation amounts, each on one line.

Output: The total amount of all donations and the rounded average donation amount, each on a separate line with a leading dollar sign and two decimal places.

Sample input:

```
39.18
45.87
22.65
19.53
45.23
60.72
```

Sample output:

```
$233.18
$38.86
```

7. Joanna

Program Name: Joanna.java Input File: joanna.dat

Joanna plans to work a summer job to earn enough money to purchase a new laptop computer and accessories before her senior year in high school. She also hopes to have some money left to spend on summer fun. With multiple job opportunities and numerous options for the computer and accessories she needs a program to evaluate the options. Joanna will not make enough to have income tax deducted from her paycheck but understands that 7.65% will be deducted for social security. Also, sales tax will be applied to the laptop and accessories purchase. She wants to know for each option how much net pay she will get working 10 weeks, how much will be spent on the laptop and accessories including sales tax, and how much will be left for summer spending.

Here is some information to understand job pay:

- Weekly gross pay = pay rate x hours worked per week
- Weekly net pay = gross pay less 7.65% (social security deduction)

Extra attention to accuracy is needed when working with money values!

Input: The first line of the data file contains a count of the number of data sets, with each data set describing a specific option to evaluate. Each of the following lines will contain one complete data set with five numeric values separated by whitespace: hourly pay rate for a job, number of weekly hours as a decimal number, price of the laptop, price of the accessories, and sales tax rate as a percentage. All values are decimal numbers except the initial count which is an integer.

Output: 3 lines of output for each set of input data followed by row of nine stars, "`*********`". Output must use exact labels as shown below with the colon, a single space, and a dollar sign preceding the value. The values are displayed as shown below with the decimal points aligned and commas as necessary. Comments in ( ) are not output.

1. `Net Pay:  $9,999.99` (total 10-week pay less social security deduction)
2. `Purchase: $9,999.99` (total price of laptop and accessories including sales tax)
3. `Fun Cash: $9,999.99` (net pay less purchase)

Sample input:

```
3
8.95 20.5 979.99 135.87 8.25
9.25 25.00 861.19 161.94 8.25
9.65 20.00 986.40 139.68 8.25
```

Sample output:

```
Net Pay:  $1,694.40
Purchase: $1,207.92
Fun Cash: $  486.48
*********
Net Pay:  $2,135.60
Purchase: $1,107.54
Fun Cash: $1,028.06
*********
Net Pay:  $1,782.40
Purchase: $1,218.98
Fun Cash: $  563.42
*********
```

9. Nina

Program Name: Nina.java **Input File: nina.dat**

Nina is the grounds keeper for all the city parks in Jayton, Texas. Recently the Jayton city council voted to create a dog park for the canine citizens of the town, naming it the 5 Star Dog Park. In the spirit of the name the council members would like for the park to be in the shape of a regular pentagon (all sides congruent, all angles congruent). There are a variety of places where the park can be built, all of which would allow for a different size park. The council would like for Nina to determine the area each potential park would cover and the amount of perimeter fencing each would require. Additionally, the council does not want to use up more than one acre of land for the new dog park. Nina has looked on the Internet and found that the area of a regular pentagon can be calculated using this formula:

$$\frac{5s^2}{4\tan\frac{\pi}{5}}$$

where s is the length of a side. The length of a side can be calculated using:

$$2r\sin\frac{\pi}{5}$$

where r is the distance from the center of the pentagon to a vertex (corner). She also found that one acre of land is 43560 square feet. Nina has come to you for help with the calculations. Write a program that will calculate the area and fencing required for each of the proposed locations for the dog park.

Input: A number N representing the number of proposed locations for the park followed by N real numbers each representing the distance in feet from the center of the proposed park to any corner of the pentagon.

Output: For each proposed location print "LOCATION #" followed by the location number followed by one space. For each location that will fit within an acre, print the area in square feet and length of fencing in feet required for the location. Each value should be rounded to the nearest hundredth and separated by one space. For any locations that will not fit within an acre print "LOCATION #" followed by the location number followed by one space and then "WILL NOT FIT".

Sample input:

```
3
100
200
75.5
```

Sample output:

```
LOCATION #1 23776.41 587.79
LOCATION #2 WILL NOT FIT
LOCATION #3 13553.15 443.78
```

10. Raghav

Program Name: Raghav.java Test Input File: raghav.dat

Raghav's math teacher gave an interesting problem with exactly 50 different integers with which to do various calculations.

The difference with this problem versus others is the students need to do the calculations to each number based on where the number is given. For instance, in the sample data below the first value is 72. Since it is first in the list, all calculations done on the number 72 are because it is the first number in the list and have nothing to do with the fact that the number is 72.

Here are the calculations and the order in which to do the calculations:

1) All even locations multiply by 2, odd locations add 7.
2) Multiples of 3 multiply by 5.
3) Multiples of 5, subtract 11.
4) Multiples of 7 square the number.
5) Multiples of 10 divide by 10.
6) Multiples of 11 find the square root.

Note: **Do not round the numbers** but instead provide the lowest integer answer.

For instance:

- Say the 33rd number in the list was 25. Since 33 is odd, a multiple of 3 and a multiple of 11, you would get the following result:

  $\sqrt{((\mathbf{25} + 7) * 5)} = 12$

- For a value 26 in position 45, which is odd, a multiple of 3, and a multiple of 5, this would be the resulting calculation and value in position 45.

  $((\mathbf{26} + 7) * 5) - 11 = 154$

- For a number in location 50 number, which might be 32, since 50 is even, a multiple of 5 and a multiple of 10, this the changed value in that position would be 5, like this:

  $((\mathbf{32} * 2) - 11) / 10 = 5$

Input: Exactly fifty integers arranged vertically in the data file, the first to be considered in location 1, an odd numbered location, and the last in position 50, which is even.

Output: Fifty integers, output in vertical format, with the mathematical changes as described above.

Sample Input (First 10 numbers only, arranged vertically in the data file)

```
72    41    25    3     24    3     12    31    11    35
```

Sample Output (First 10 numbers only, to be output in vertical format)

```
79    82    160   6     20    30    361   62    90    5
```

1. Alfonso

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

Alfonso was playing with for loops and strings and came up with the following pattern. Can you recreate it with for loops? Do you need to hard code it?

Input: None

Output: Write a program that displays the pattern you see below.

Exact Output Expected:

```
ACEGIKMOQSUWY
CEGIKMOQSUWY
EGIKMOQSUWY
GIKMOQSUWY
IKMOQSUWY
KMOQSUWY
MOQSUWY
OQSUWY
QSUWY
SUWY
UWY
WY
Y
EGIKMOQSUWY
GIKMOQSUWY
IKMOQSUWY
KMOQSUWY
MOQSUWY
OQSUWY
QSUWY
SUWY
UWY
WY
Y
IKMOQSUWY
KMOQSUWY
MOQSUWY
OQSUWY
QSUWY
SUWY
UWY
WY
Y
MOQSUWY
OQSUWY
QSUWY
SUWY
UWY
WY
Y
QSUWY
SUWY
UWY
WY
Y
UWY
WY
Y
Y
```

2. Barb

Program Name: Barb.java **Input File: barb.dat**

Barb, a math teacher, has decided to supply all her students with a chart of simple math formulas and values to help them with their work. She only wants to provide a chart up to a certain number.

Each row of this chart will include a number, the square of the number, 3 times the number, and half of the number (no fractional part, just the whole number answer). The chart will start at 1 and continue up to a certain number Barb decides. Each chart value will be separated by exactly 5 spaces. The first line will be the chart heading which will have exactly 3 spaces between each item in the heading.

```
A   A*A   3*A   A/2
```

All numbers will be displayed as integers.

Input: A single integer N (0 < N < 100).

Output: A math chart as shown in the example below and formatted exactly as described above.

Sample Input:

```
10
```

Sample Output:

```
A   A*A   3*A   A/2
1     1     3     0
2     4     6     1
3     9     9     1
4     16     12     2
5     25     15     2
6     36     18     3
7     49     21     3
8     64     24     4
9     81     27     4
10     100     30     5
```

10. Nisha

Program Name: Nisha.java **Input File: nisha.dat**

Nisha just learned about writing numbers in binary in class. She loves powers of two, and often doodles during class. Her most recent doodles are lists of binary representations of numbers. Nisha doesn't consider a list completed until she's written out every number from 1 to 2^k (inclusive) for some positive integer k. For example, here's her list for k =3.

```
1
10
11
100
101
110
111
1000
```

Nisha was curious how many times she writes the digit 1 as part of this list. Help her find the answer by writing a program.

Input: The first line is T (at most 20), the number of test cases to follow. Each test case contains a single positive integer k (at most 50). This means that Nisha's list contains the binary representations of all numbers between 1 and 2^k (inclusive).

Output: For each test case, print the case number and the total number of times the number 1 appears in the binary expansion of all integers between 1 and 2^k (inclusive), formatted as in the sample. Note the answer may not fit into a 32-bit integer.

Sample input:

```
3
3
1
15
```

Sample output:

```
Case #1: 13
Case #2: 2
Case #3: 245761
```

12. Vlad

Program Name: Vlad.java Input File: vlad.dat

Vlad is the chairman of the recruitment committee for the Garden Club at Carlton High School. Go Bulldogs! The committee has decided to place an emphasis on recruiting freshman students to join the club this year. As chairman he has taken the responsibility of obtaining a list of the names of all the freshmen in CHS and dividing that list amongst the committee members so that each freshman is contacted and asked to join. The list of freshmen the registrar provided Vlad has each student listed with their student ID number, last name, first name and middle initial.

He would prefer the list he provides his committee members just be first name, middle initial and last name in alphabetical order by last name. That's where you come in. Write a program that will read the list provided by the registrar and print a list where each student is shown as first name, middle initial and last name in alphabetical order based on last name, then first name and finally by middle initial.

Input: A list of unknown size containing the ID numbers and names of each of the freshman students in Carlton High School each on a separate line. The names will be shown as last name followed by a comma and exactly one space then first name followed by one space and a middle initial followed by a period. Some students might not have a middle initial.

Output: A list of freshman students that is alphabetized by last name then first name and finally by middle initial. The names should be printed first name followed by one space, middle initial followed by one space and then last name. There should be no punctuation within the names and each name should be on a separate line. A name containing no middle initial takes precedence over the same name with an initial.

Sample input:

```
196 Perez, Ava P.
205 Hill, Scarlett M.
369 Gonzalez, Olivia G.
1025 Rodriguez, David D.
1170 Walker, Alexander E.
1401 Davis, Thomas Z.
1492 Clark, Ximena K.
2515 Smith, Daniel B.
2747 Anderson, Jayden F.
2827 Lewis, Jose G.
```

Sample output:

```
Jayden F Anderson
Ximena K Clark
Thomas Z Davis
Olivia G Gonzalez
Scarlett M Hill
Jose G Lewis
Ava P Perez
David D Rodriguez
Daniel B Smith
Alexander E Walker
```