SPRING 2019 ACM PROGRAMMING CONTEST

Overview

Welcome to the Spring 2019 ACM Programming Competition! This is a contest wherein you compete amongst your fellow students, and professors, in an effort to solve a set of programming problems as fast as possible.

Problems

There are nine problems of (roughly) increasing difficulty. The team who solves the most problems is declared the winner. In the event of a tie, the team with the best time wins. The best time also includes penalties for wrong submissions. Each problem has a general description, details on the formats for input and output, and example input and output.

Input is done from standard input and output is done to standard output. *Do not open any files inside of your program.* Output of your program must match that of the correct output *exactly*. You can solve the problems in C, C++, Java, Python, or Julia. Up to date versions of these languages will be used to judge your entries. You should save your code using the standard extension for your language of choice (.c .cpp .java .py and .jl respectively).

At the end of your packet, there is also a brief overview of how to perform input and output in C++, Python and Java in case you need a refresher.

Rules

You are allowed to have any printed material such as books or printouts. You are also allowed to access standard library documentation for your language of choice. *No other use of the internet is permitted.* You are also not allowed to copy and paste code from any source be it the web, a thumb drive, or another file that you have. Only one computer is allowed to a team.

Logging In

To log in to the system, direct your browser to `https://bit.ly/2D2yjBc`. You will then need to enter your team name and password which are provided for you. This will bring up the interface for you to submit solutions to the problems, ask questions (which will be broadcast to all participants), and see how your team is doing.

Submitting

To submit a solution, first make sure that you select the correct problem letter on the top of the site. Then click the “Choose File” button. Then choose the file that contains your source code (be sure it has the correct extension). After that it will be submitted for judging. It will receive an automatic judging response which is marked pending. Your solution will then be checked manually and a final response will be assigned.

DO NOT TURN OVER THIS SHEET UNTIL THE CONTEST STARTS

Problem A: Bagging Ducks

In Duck Hunt, players attempt to shoot ducks in mid-flight as they fly across the screen. When two players are playing, the two players both try to shoot as many ducks as they can. Whoever shoots the most is the winner. If both players shoot an equal amount, it is a tie. Write a program which reads in the number of ducks each player bagged, and print out the winner, or indicate that the game ended in a tie.

Input

Input will consist of two lines. On the first line is an integer giving the number of ducks that player 1 shot. On the second line is an integer giving the number of ducks that player 2 shot.

Output

You should output one line saying either “Player 1 is the winner!”, “Player 2 is the winner!” or “It is a tie.”, based on which player bagged more.

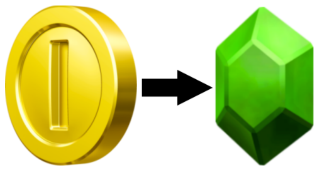
Sample Input

```
13
9
```

Sample Output

```
Player 1 is the winner!
```

Problem B: How Many Rupees?



Mario and Luigi are travelling to the land of Hyrule to compete in a Smash competition being held in a stage in Hyrule Castle. They need to pick up a few things at a shop, for the upcoming match.

Unfortunately, they only have the coins which are the currency in Mushroom Kingdom. They need to exchange these coins for the Rupees which are the main currency throughout Hyrule.

The brothers stop at Chudley's Fine Goods and Fancy Trinkets Emporium which also functions as a currency exchange for travellers. Chudley exchanges 1 Rupee for every 3 Mushroom Kingdom coins. He also takes 3 Rupees as an exchange fee off the top. Any remaining coins he simply keeps.

For example, 37 coins will be exchanged for 12 Rupees, with 1 coin leftover. Chudley keeps the leftover coin, and 3 Rupees, giving 9 out to the travellers.

You will write a program that will figure out how many coins Mario and Luigi get back based on how many coins they have to exchange.

Input

Input will contain 1 line, giving the number of coins the brothers are exchanging.

Output

Output will contain 1 line, saying "*X* Rupees" where *X* is the number of Rupees the brothers receive back.

Sample Input

```
37
```

Sample Output

```
9 Rupees
```

Problem C: Blue Shells

In Mario Kart races, competitors vie to see who can complete three laps around a track the fastest. Along the way, the racers collect power-ups which help them go faster or harm their competitors.

One such power-up is the blue turtle shell. This projectile, when fired, will seek and out smash into every racer who is ahead of the racer that fires it. For example, if there are 4 racers, and the kart in 3rd place fires the shell, it will smash into the kart in 2nd place, then the one in 1st place, but leave the kart in 4th place alone.

You must write a program that prints out the list of racers who will be hit by the blue shell when it is fired by a particular racer.

Input

The first line of input will be the name of the racer who fires the blue shell. The second line will be an integer, N, giving the total number of karts in the race. Following that will be N lines, each giving the name of a racer, in order. The first of these lines is the competitor currently in 1st place, and the last is the competitor currently in last place.

Output

Your program should output a message for each kart hit by the blue shell. You should print them out in the order they will be hit (closest to the kart which fires it though the kart in 1st place).

Sample Input

```
Luigi
8
Donkey Kong
Princess Peach
Yoshi
Mario
Bowser
Luigi
Koopa Troopa
Toad
```

Sample Output

```
Bowser is hit!
Mario is hit!
Yoshi is hit!
Princess Peach is hit!
Donkey Kong is hit!
```

Problem D: Paperboy Scoring

The Daily Sun employs paperboys to deliver their newspaper in a suburban town. These paperboys must deliver papers to all subscribers, without damaging their homes. Also, the Daily Sun provides bonuses to paperboys who vandalise the homes of non-subscribers. Such is the cutthroat world of the suburban news business.

The Sun has asked you to write a program which will give the paperboys a score based on the following factors:

Code	**Action**	**Score**
MB	Delivery in subscriber mailbox	+15
PO	Delivery on subscriber porch	+7
YA	Delivery on subscriber yard	+5
BW	Broken subscriber window	-25
NS	Delivery to non-subscriber	-15
NW	Broken non-subscriber window	+8
ST	Stealing of competing paper	+12

The paperboy receives the total score of all actions the perform. For example, if a paperboy delivers the paper to a subscriber's mailbox twice (+30), breaks a non-subscriber's window (+8), and breaks a subscriber's window (-25), then their total score is 13.

Input

The first line of input will be an integer N. Following that will be N lines each containing one of the codes above. Each time a code appears in the input corresponds to one instance of the paperboy performing that action.

Output

You should output the total score for the paperboy in the format "Score is X." where X is the total score.

Sample Input

```
10
PO
YA
NS
MB
BW
YA
NW
ST
MB
YA
```

Sample Output

```
Score is 32.
```

Problem E: The Banana Hoard

In their travels across Kong Isle, five members of the Kong family come across a large hoard of bananas. The five members (Donkey Kong, Cranky Kong, Dixie Kong, Diddy Kong and Funky Kong) decide to share the bananas evenly. As it is getting late, they decide to go to sleep and split the bananas the next morning.

During the night, however, Donkey Kong wakes up. He decides he doesn't trust the other apes and wants his share early. He sneaks over to the pile of bananas and takes $\frac{1}{5}$ of them for himself. If the pile doesn't divide evenly by 5, he eats the remaining banana(s). He sneaks back to his sleeping area with this share of bananas.

Later in the night, Cranky Kong wakes up and does the same thing. He sneaks up to the pile and takes $\frac{1}{5}$ of whatever Donkey Kong left back to where he is sleeping. If the pile doesn't divide evenly by 5, he also eats the remainder.

The other three members of the Kong family, (Dixie, Diddy and Funky), do the same thing, in that order.

When morning comes, the Kongs evenly divide the (much smaller) pile of bananas into 5 equal shares. Each Kong adds this share to the bananas they stole over the night. If any bananas are remaining when this pile is divided by 5, they bake them into a banana creme pie.

Input

Input will consist of a single line, giving an integer N, the number of bananas the Kongs start with. N will be at least 25 and no more than 1,000,000.

Output

Your program should output how many bananas each member of the Kong family received total, along with how many they secretly ate. You should print the totals for the Kongs in the same order as they woke up in. The last line of output should state how many bananas were baked into the pie.

Sample Input

```
100
```

Sample Output

```
Donkey Kong has 25 bananas, and ate 0.
Cranky Kong has 21 bananas, and ate 0.
Dixie Kong has 17 bananas, and ate 4.
Diddy Kong has 14 bananas, and ate 3.
Funky Kong has 12 bananas, and ate 1.
3 bananas were baked into a pie.
```

Problem F: Ranking the Fighters

Toad has recently been put in charge of ranking combatants who fight in the Super Smash Bros contests. In this particular competition, there is no playoff bracketing, just a series of head to head matches between players. Each match has a clear winner and loser.

Toad needs to rank the players based on how well they did in these individual matches. After deliberating for some time, Toad has decided that he will use the following criteria to rank the players:

- Any combatants who have not lost should be ranked first. If there are multiple with zero losses, they should be sorted based on the number of wins they have – with the highest number being ranked first. If some have the same number of wins, they should be ranked alphabetically.

- Any combatants who have not won should be ranked last. If there are multiple with zero wins, they should be sorted based on the number of losses they have – with the lowest number being ranked first. If some have the same number of losses, they should be ranked alphabetically.

- For players with both wins and losses, they should be given a score based on their record. Each win adds 2 points to their score. Each loss deducts a point. So if a player has 3 wins and 2 losses, their score would be 4. These players should be ranked from highest score to lowest. In the event of ties, use player names as a tie-breaker.

If any combatants have not participated in any matches at all, they should not appear in the ranking list.

Input

The first line of input gives the number of combatants, N. Following that are N lines giving the name of the combatants.

Next there is a line giving the number M of matches. Following this line are M lines giving the outcome of the matches. These lines have the winning player's name, the word "beats", then the losing player's name, and end in a period.

Output

Your program should output the ranked list of fighters using Toad's scheme. There will be 1 line for each combatant who participated in at least one match. For each line, give the name of the player, then a colon and space. Next give the number of wins and losses they have, separated with a hyphen.

Sample Input

```
11
Mario
Sonic
Link
Pikachu
Luigi
Samus
Falco
Bowser
Kirby
Yoshi
Ike
18
Falco beats Pikachu.
Falco beats Sonic.
Link beats Ike.
Pikachu beats Ike.
Luigi beats Falco.
Link beats Falco.
Falco beats Luigi.
Ike beats Pikachu.
Mario beats Bowser.
Link beats Luigi.
Ike beats Pikachu.
```

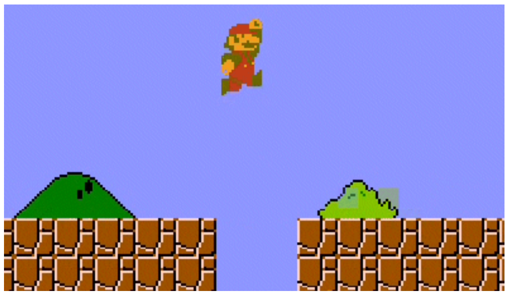
```
Luigi beats Yoshi.
Falco beats Yoshi.
Samus beats Falco.
Ike beats Pikachu.
Sonic beats Samus.
Sonic beats Ike.
Samus beats Sonic.
```

Sample Output

```
Link: 3-0
Mario: 1-0
Falco: 4-3
Ike: 3-3
Samus: 2-1
Luigi: 2-2
Sonic: 2-2
Pikachu: 1-4
Bowser: 0-1
Yoshi: 0-2
```

Problem G: Platform Puzzle

In his travels, Mario often has to jump across a series of platforms from a starting point to an ending point. Some platforms he can get across and some he can't (for those he can find an item such as a feather or leaf which will allow him to fly). Your job is to help Mario by writing a program which will determine if he can or cannot traverse a series of platforms.

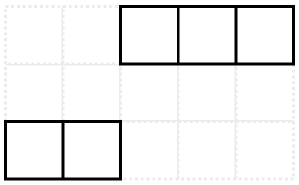


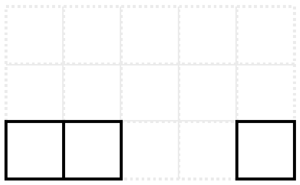
A level consists of a two-dimensional grid of blocks. Mario begins on the bottom-left side and must get to the far right side. The level consists of blocks, but also gaps. Mario can stand or walk on blocks and can jump from one block to another. Mario's movement follows these rules:

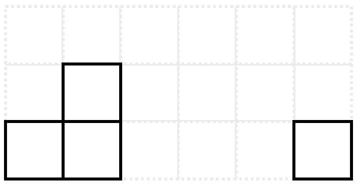
1. Mario can only move to the right, he can never go back to the left.
2. To be able to jump, Mario must not have a block directly above his head. He can jump 2 blocks up from his current position. He can also choose to jump just 1 block up, or choose to walk off the edge of a block.
3. If Mario walks off the edge of a block, and there is a block adjacent, he will be on that block. If there is not, he will be in mid-air.
4. When Mario is in mid-air (either because he walked off the edge of a block, or because he jumped), he will begin to fall. For each position he falls, he is able to move 1 position to the right. He can also choose to fall straight down if he wants to.
5. Mario must be able to stand on a block between jumps/falls. There also must always be a space above the block that Mario is standing on.
6. Mario is able to jump off the top of the level. However, if he falls past the bottom row, then he dies.

Examples

Following these rules, Mario *can* make the following three jumps:

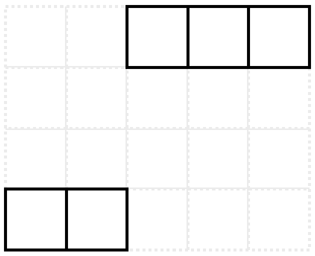

(a)

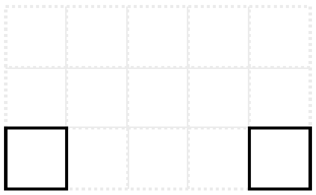

(b)

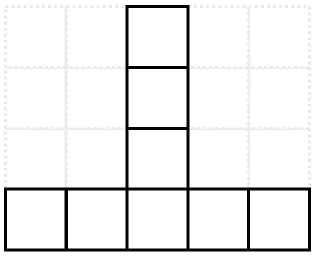

(c)

In (a), Mario can jump up two rows, and then land on the adjacent bock to get to the platform on the right. In (b), Mario will need to jump up two levels as well. He then goes to the right two spaces, dropping two rows each time, to land on the next platform. In (c) Mario clears a gap of three spaces. He can do this with a jump of size 2, and because the next platform is one block down from his take-off point.

The following are examples of levels Mario *cannot* clear:


(d)


(e)


(f)

The problems in (d) and (f) are both that Mario cannot jump 3 rows up, only 2. For (d) this means he can't reach the upper platform. For (f) it means he can't get across the blocks in his way. For (e) the problem is that Mario is unable to cross a gap of size three, because he will lose one block of height for each block the gap is.

Input

The first line of input contains an integer, R, giving the number of rows in the level. The second line contains an integer, C, giving the number of columns in the level. Following that are R rows each with C characters.

Each character is either a # character or a space. The # character indicates a solid block which Mario can walk on (but not through). The space characters indicate a gap where no block exists.

Output

Your program must output one line containing either “Mario can make it!” or “Mario cannot make it!”.

Sample Input 1

```
3
16
        #

####  ###    ###
```

Sample Output 1

```
Mario can make it!
```

Sample Input 2

```
3
16

       #
#####  #########
```

Sample Output 2

```
Mario cannot make it!
```

Problem H: Tricky Switches

Link is navigating his way through a dungeon, looking for one of the Spiritual Stones which will allow him to enter the Sacred Realm and claim the Triforce before Ganondorf can use it to increase his power.

On the way, Link comes upon a row of switches which work in an interesting way. Whenever a switch is flipped, the switches next to it are automatically flipped as well.

For example, suppose there are 4 switches, and they all start in the off position:

1	**2**	**3**	**4**
Off	Off	Off	Off

Now suppose that Link pulls switch 1. That will turn that switch on and also flip the adjacent switch (number 2) to on as well:

1	**2**	**3**	**4**
On	On	Off	Off

If he pulls switch 3 next, then it will be on. It will also flip switches 2 and 4 because they are adjacent, giving this:

1	**2**	**3**	**4**
On	Off	On	On

In order to pass through the dungeon, Link needs to get all switches into the On position. He also wants to do so by pulling as few of the switches as possible.

You will write a program that will be given the number of switches, along with their initial configuration. From that you must determine the minimum number of pulls required to get all the switches on. Some initial configurations have no possible solution. In that case, your program should let Link know the puzzle is impossible.

Input

The first line of input is an integer N, such that $2 \leq N \leq 10$. Following this will be N lines which give the initial state of the switches. Each line will contain either the word “On” or “Off”. The first line corresponds to switch 1, the second to switch 2 and so on.

Output

Output should consist of 1 line. If Link can solve the switch puzzle, this line should say "Link must pull K switches.", where K is the minimum number of switches needed to set all switches to On. If there is no solution, the line should say "There is no solution."

Sample Input 1

```
3
On
Off
On
```

Sample Output 1

```
Link must pull 3 switches.
```

Sample Input 2

```
2
Off
On
```

Sample Output 2

```
There is no solution.
```

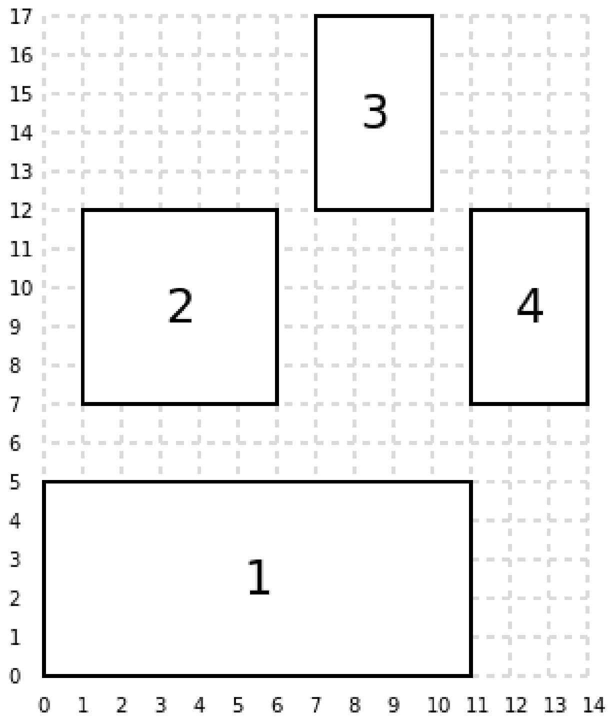
Problem I: Set the Laser

Fox McCloud is tailing one of Andross's frigates in his Arwing fighter. The frigate, which is a larger ship with multiple rooms, is launching an attack against a peaceful world in the Lylat system, so McCloud needs to bring it down.

The Arwing is equipped with a powerful laser, the blast from which does severe damage to anything in its path. In order to do the most damage to the large frigate, the laser needs to destroy as many rooms in the frigate as possible.

The Arwing's computer is able to download a schematic of the frigate. This schematic will give the location of each room inside the ship. Your task is to help McCloud find out how many of the rooms his laser can hit, based on the layout of the rooms.

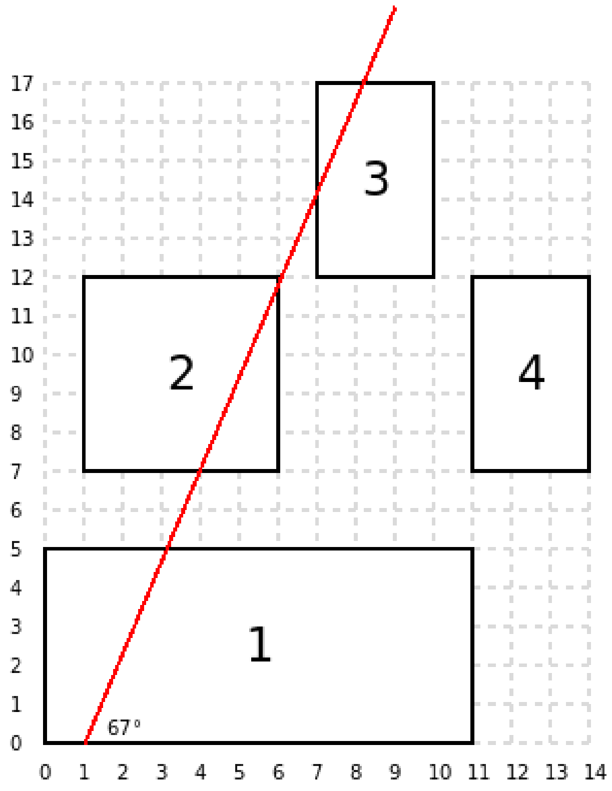
For example, consider the room layout below. This is the test case in the Sample Input below.



Here there are four rooms. As in this example, the rooms will always consist of rectangles, which are axis-aligned (this means no rooms will be angled at all). The room coordinates will always be integers.

Because McCloud is behind the frigate, the laser will always enter from the bottom of the schematic. To configure the laser, McCloud must give the X coordinate to shoot from, and the angle of the laser. The X coordinate of the laser is a real number between 0 and the width of the frigate inclusive (so from 0 to 14 in this case). The angle can be any real number greater than 0°and less than 180°.

McCloud wants to find the configuration which will hit a maximum number of rooms. In this example, it's possible to hit three rooms, but not all four. Here is a possible setting to hit three:



Input

The first line of input gives the number of rooms, N, which will be between 1 and 100 inclusive. Following this are N lines giving the locations of the N rooms. Each line consists of 4 numbers, separated by spaces: X_1 Y_1 X_2 Y_2. The coordinates (X_1, Y_1) give the lower left-hand corner of the room. The coordinates (X_2, Y_2) give the upper right-hand corner of the room.

All coordinate values will be integers between 0 and 100 inclusive. No rooms will ever be touching, not even at a single point, and no rooms can have 0 area. The left-most room will always be at $X = 0$ and the bottom-most room will always be at $Y = 0$.

Output

The output should be a single line saying “We can hit K rooms!” where K is the maximum number of rooms the laser can hit. If there is only 1, you should print “We can hit 1 room!”.

Sample Input 1

```
4
0 0 11 5
1 7 6 12
7 12 10 17
11 7 14 12
```

Sample Output 1

```
We can hit 3 rooms.
```

Doing Input and Output

This guide contains the basic way of doing input and output in C++, Java, and Python, in case you need a quick refresher.

In programming contests, you should always do input from the terminal screen (e.g. using the keyboard), and do output to the terminal screen. You should not open any files for input or output.

You should also not output any prompts. If a problem instructs you to read a line containing a number, just read it in without any kind of prompt saying “Enter a number” or similar.

C++

Input

1. To skip over whitespace (spaces or new lines), and read in a single value (such as an integer or string), use `cin` as follows:

   ```
   // read one integer
   int number;
   cin >> number;

   // read one character string
   char str1[100];
   cin >> str1;

   // read one string object
   string str2;
   cin >> str2;
   ```

2. To read in one entire line of input, which may contain spaces, use `cin.getline`. For example:

   ```
   // read in a character string of up to 100 characters
   char str1[100];
   cin.getline(str1, 100)

   // read in a string object
   string str2;
   getline(cin, str2);
   ```

Output

Output is done with `cout` in C++ which can take any built-in data type. For example:

```
// print a message, then an integer, then a new line
int x;
cout << "X is equal to " << x << endl;
```

Python

Input

1. The `input` function in Python read in one line of input and returns it as a string. For example:

   ```
   # read in a string
   line = input()
   ```

2. In order to convert from a string to a number, you can use the `int` function for integers, or the `float` function for real numbers:

   ```
   # read in an integer by passing the input string to int()
   number = int(input())
   ```

3. In order to break a line of input into multiple strings, separated by spaces, you can use the `.split()` function which returns a list of strings:

   ```
   # read in a whole line
   line = input()

   # split it into separate strings based on spaces
   words = line.split()
   ```

Output

1. Output in Python is done with the `print` function which outputs all of its arguments, separated by spaces, and puts a newline at the end:

   ```
   # print a message, a number and a new line
   print("X is equal to", x)
   ```

2. In order to prevent `print` from putting spaces between each item, pass `sep=' '` as an argument:

   ```
   # now there is no space between message and value
   print("X is equal to", x, sep= )
   ```

3. In order to prevent `print` from putting a new line at the end, pass `end=' '` as an argument:

   ```
   # now there is no new line added
   print("X is equal to", x, end= )
   ```

Java

Input

Input in Java can be done with the `Scanner` class which must be imported first:

```
import java.util.Scanner;
```

Then a `Scanner` object must be created:

```
java.util.Scanner in = new java.util.Scanner(System.in);
```

1. To read in one line of input into a string, use the scanner's `nextLine` method:

   ```
   String line = in.nextLine();
   ```

2. To read in a single word into a String, stopping at a space, use the `next` method:

   ```
   String word = in.next();
   ```

3. To read a numerical value use the `nextInt` method for integers, or the `nextDouble` method for real numbers:

   ```
   int number1 = in.nextInt();
   double number2 = in.nextDouble();
   ```

Output

1. To output a string constant or variable, use the `System.out.println` function which takes one argument, prints it to the screen, then prints a new line:

   ```
   // print a message
   System.out.println("This will be printed")

   // print a value
   int x;
   System.out.println(x);
   ```

2. To output something without a new line at the end, use the `System.out.print` function which outputs its argument with no newline:

   ```
   // print a message with no new line
   System.out.print("The value of X is ")

   // print a value with no newline
   int x;
   System.out.print(x);
   ```