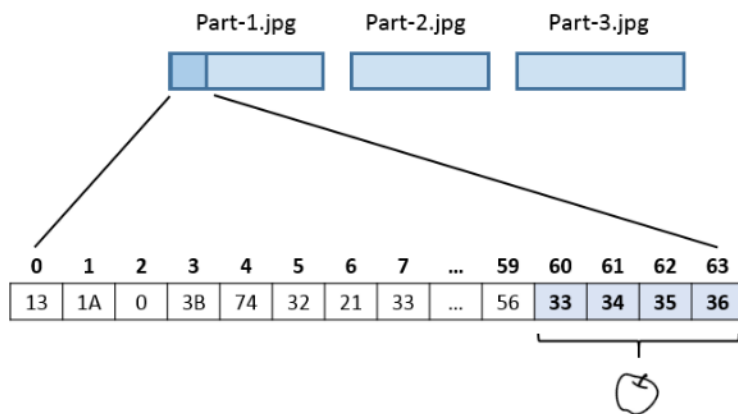


## Problem 3 - Hidden Secrets

Have you heard about the [Wow! signal](#)? We've just received an even bigger signal from one of our satellites. We've managed to extract the information, but just minutes before the breakthrough, the CIA marched in and wiped out our team. Now everything lies in your hands - find the secrets that lie within the signal!

This is what we know so far: We have **3 files** and each file is separated into blocks of **64 bytes**. The last **4 bytes** of each block hold a **holy sign**. A holy sign can either be an **apple**, a **leaf** or a **cross**. The bytes corresponding to each sign are listed below.

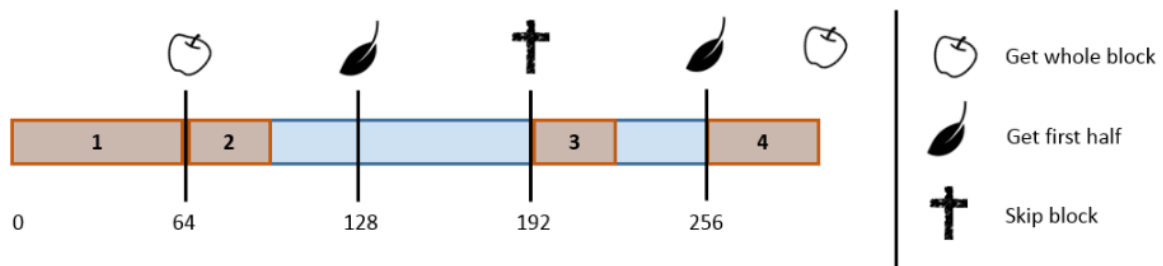


Sign	Bytes (in hex)
Apple	33 ('3'), 34 ('4'), 35 ('5'), 36 ('6')
Leaf	21 ('!'), 22 ('"'), 23 ('#'), 24 ('\$')
Cross	2B ('+'), 2C (','), 2D ('-'), 2E ('.')

Your task is to assemble the 3 files into one. We know for certain that:

- Blocks **marked with an apple** should be included in the result file.
- Blocks ending with a **leaf sign** should only have their **first half** included in the result.
- The **cross sign**, on the other hand, denotes the memory is garbage and should be **skipped**.

The example below illustrates how we **extract memory** from each part.



Note that the last 4 bytes should **NOT** be included when calculating the size.

For example, the last block of a part may have less than 64 bytes. If a **leaf block** is 9 bytes long, 4 bytes are reserved for the sign; we're left with 5 -> **first half** is 2).

The program should work with any file extensions.

### Input

The input will be read from the terminal. The input parameters will be the **source file names**. They will all have the **same extension**. They will be space separated. See the examples given below.

### Output

The extracted data should be written to a result file with the **same extension as the input files**.

- In case of success, print **"Eureka!"** at the end of the program
- In case of missing input files, print **"Usage: [<src-file-1> <src-file-2> ...]"**
- In case of invalid file names, print **"{file-name}: No such file or directory"**

## Constraints


- Each block is guaranteed to have at least 4 bytes. Each block will end with a sign.
- The program should display **no memory leaks, buffer overflows or dangling pointer anomalies**.
- Using C++ is **forbidden**.

## Examples

### Test #1

Input from Terminal		Output
./program text-1.txt text-2.txt		Eureka!
text-1.txt	text-2.txt	merged.txt
AAAAAAAAAAAAAAAAAAAAAA AAAAAAAAAAAAAAAAAAAAAA AAAAAAAAAAAAAA3456BBBBB BBBBBBBBBBBBBBBBBBBBBB BBBBBBBBBBBBBBBBBBBBBB BBBBBBBBBB+, - .CCCCCCCCCCCCCCCCCCCC CCCCCCCCCCCCCCCCCCCC CCCCCCCCCCCCCCCC!"#\$DDDD D!"#\$	PPP3456	AAAAAAAAAAAAAAAAAAAAAA AAAAAAAAAAAAAAAAAAAAAACC CCCCCCCCCCCCCCCCCCCCCDDP PP

### Test #2

Input from Terminal			Output
./program Part-1.jpeg Part-2.jpeg Part-3.jpeg			Eureka!
Part-1.jpeg	Part-2.jpeg	Part-3.jpeg	merged.jpeg
(binary)	(binary)	(binary)	

### Test #3

Input from Terminal			Output
./program			Usage: [<src-file-1> <src-file-2> ...]
-	-	-	-

## Test #4

Input from Terminal			Output
./program Part-1.jpeg Part-2.jpeg Part-3.jpeg			Part-1.jpeg: No such file or directory
Part-1.jpeg	Part-2.jpeg	Part-3.jpeg	-
(does not exist)	(binary)	(binary)	