**Ko Bancha Coder (KBC)  
Contest Problems (Sorted by Difficulty)**

|  |  |  |
| --- | --- | --- |
| Problem | Difficulty | Points |
| The Time Converter | Easy | 25 |
| Secret Society Cipher – Alphabet Key Edition | Medium | 50 |
| Advanced Inventory Batch-Forming | Hard | 75 |

# Problem 1 (25 Points – Easy): The Time Converter

**Scenario:**  
You're developing a smart display system for an international airport. The time inputs are provided in various 24‑hour styles, and you must convert them into clean, readable 12‑hour AM/PM format. The system should handle multiple valid 24‑hour input styles and format the output appropriately.

* **Valid Input Styles:**

|  |  |
| --- | --- |
| **Style** | **Example** |
| HH:MM | 13:45 |
| H:MM | 9:05 |
| HHMM | 2210 |
| HMM | 930 |

**Input:**  
A single time string in one of the formats above (00:00 – 23:59).

**Output:**  
The corresponding 12‑hour time in the format “H:MM AM” or “H:MM PM”.  
Hours must not have a leading zero.

* **Examples:**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 13:45 | 1:45 PM |
| 9:05 | 9:05 AM |
| 00:10 | 12:10 AM |
| 9:30 | 9:30 AM |
| 22:10 | 10:10 PM |

# Problem 2 (50 Points – Medium): Secret Society Cipher – Alphabet Key Edition

**Scenario:**  
In an underground hacker society, members encrypt messages using a powerful string cipher. Your task is to implement this encryption based on strict rules and compute a final alphabet key.

* **Encryption Rules:**

The input string contains only alphabets (a–z, A–Z) and spaces.

Split the input into words using spaces.

For each character in every word:

– Map it to its reverse position in the alphabet (a or A → 26, …, z or Z → 1).

– If the character is a vowel (a, e, i, o, u), square this value.

– If the character is uppercase, add 10 to the final value (if both uppercase and vowel, square first, then add 10).

Separate two words corresponding values with a pipe “|” like shown in output below.

Compute the sum of all numeric values, take modulo 26 then Calculate “-KEY: [X]” where X is the derived letter from modulo value, mapped using: 0 → A, 1 → B, ..., 25 → Z. The key should be printed in \*\*uppercase\*\*.

**Input:**  
A single line string S (1 ≤ |S| ≤ 200).

**Output:**  
<encoded | words>-KEY: [X]

* Example (with step‑by‑step trace):

**Input:** Rise Above

Word 1 – "Rise":

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Char | Reverse Pos | Vowel Squared? | Uppercase +10? | Final Value(Total Addition) |
| R | 9 | No | +10 | 19 |
| i | 18 | 18² = 324 | No | 324 |
| s | 8 | No | No | 8 |
| e | 22 | 22² = 484 | No | 484 |

Word 2 – "Above":

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Char | Reverse Pos | Vowel Squared? | Uppercase +10? | Final Value |
| A | 26 | 26² = 676 | +10 | 686 |
| b | 25 | No | No | 25 |
| o | 12 | 12² = 144 | No | 144 |
| v | 5 | No | No | 5 |
| e | 22 | 22² = 484 | No | 484 |

**Total Sum =** 2179

**2179 % 26 = 21**

Mapping 21 → V (using 0 → A, …, 25 → Z).

**Final Output:** 19 324 8 484|686 25 144 5 484-KEY:V

* **Examples:**

|  |  |
| --- | --- |
| **Input** | **Output** |
| Hetauda City College | 29 484 7 676 36 23 676|34 324 7 2|34 144 15 15 484 20 484-KEY:K |
| HELLO | 29 494 25 25 154-KEY:Z |
| We are Legion | 14 484|676 9 484|25 484 20 324 144 13-KEY:Z |
| God Is Great | 30 144 23|334 8|30 9 484 676 7-KEY:D |
| Medium level QUESTION | 24 484 23 324 36 14|15 484 5 484 15|20 46 494 18 17 334 154 23-KEY:Y |

# **Problem 3 (75 Points – Hard): Advanced Inventory Batch-Forming**

## Scenario:

You are tasked with automating shipment formation in a high‑tech warehouse. The warehouse stocks products labelled with IDs 1 … n. Shipments ("batches") must comply with strict numerical and security rules.

## Rules & Workflow:

1. 1. Initial pruning (k‑spaced removal).

   • Write the IDs 1 … n in order. Starting from the first position (index 0), count k positions forward \*in the current list\* and remove that item. Repeat the count‑and‑remove process until you reach the end of the list once.  
   • For example, with n = 10 and k = 3, the removals are ID 4 and then ID 8. (See trace below)

1. 2. Batch size. A batch must contain exactly k distinct remaining IDs (elements in batches should be in ascending order).
2. 3. No‑multiple rule. Inside one batch, no ID is a multiple of another.
3. 4. Prime‑sum rule. The sum of the IDs in a batch must be a prime number.
4. 5. Genres for statistics. A product’s genre is the first letter of its English name (1 → “one” → o, 2 → “two” → t, …). Genres are not a constraint — they are counted only for the final answers.

## Input:

n k

## Output:

• Number of valid batches

• Number of unique genres that appear in those batches

• The set of those genre letters

## Corrected Example (step-by-step for n = 10, k = 3)

Input:  
10 3

### Step 1 – k-spaced removal (k = 3):

Start with the list:  
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

• Start at index 0 → count 3 ahead → index 3 → remove 4   
• New list: [1, 2, 3, 5, 6, 7, 8, 9, 10]   
• Resume counting from index 3 → count 3 ahead → index 6 → remove 8

➡️ Removed IDs: 4, 8

➡️ Remaining IDs: 1, 2, 3, 5, 6, 7, 9, 10

### Step 2 – Evaluate every possible batch of size k = 3

We now check:  
- Each combination of 3 IDs  
- Whether they satisfy the no-multiple rule  
- Then check if their sum is prime  
- Finally, extract genres from valid batches

### Valid Batches (after applying all rules):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Batch | Sum | Prime? | No-Multiple? | Genres |
| {2, 5, 10} | 17 | Yes | Yes | t, f, t |
| {2, 7, 10} | 19 | Yes | Yes | t, s, t |
| {3, 5, 9} | 17 | Yes | Yes | t, f, n |
| {6, 7, 10} | 23 | Yes | Yes | s, s, t |

### Final Valid Batches (that pass all rules):

• {2, 5, 10} → genres = {t, f}  
• {2, 7, 10} → {t, s}  
• {3, 5, 9} → {t, f, n}  
• {6, 7, 10} → {s, t}

### Genres:

All unique genres = {t, f, s, n}

## Final Output:

Number of valid batches = 4

Number of unique genres = 4

Genres = {t, f, s, n}

* **Example Testcases:**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 10, 3 | Number of valid batches = 1, No of unique genres = 4, The genres = {s, t} |
| 6, 2 | Number of valid batches = 1, No of unique genres = 2, The genres = {f, t} |
| 5, 2 | Number of valid batches = 1, No of unique genres = 0, The genres = {f, t} |
| 15, 3 | Number of valid batches = 14, No of unique genres = 4, The genres = {e, f, n, s, t} |
| 3, 3 | Number of valid batches = 0, No of unique genres = 0, The genres = {} |