Cryptopangrams (10pts, 15pts)

Problem

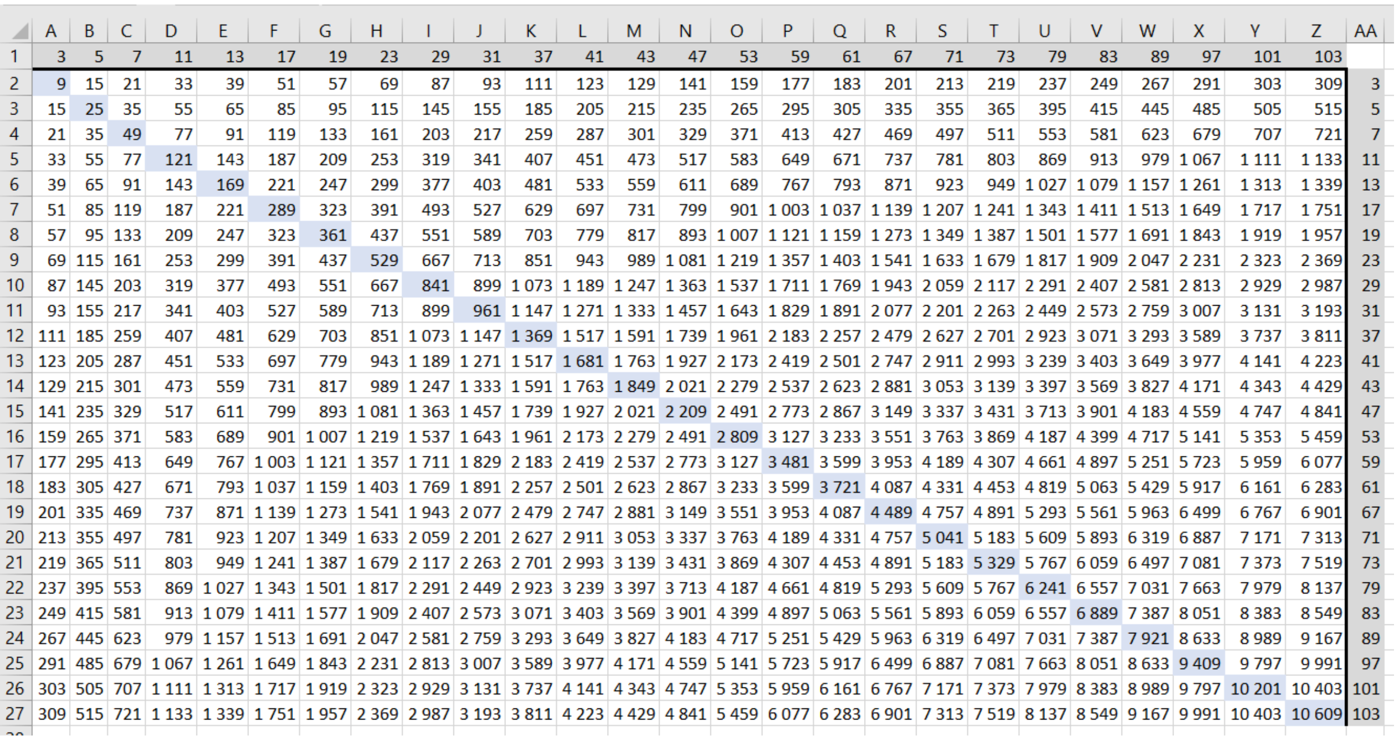
On the Code Jam team, we enjoy sending each other *pangrams*, which are phrases that use each letter of the English alphabet at least once. One common example of a pangram is "the quick brown fox jumps over the lazy dog". Sometimes our pangrams contain confidential information — for example, `CJ QUIZ: KNOW BEVY OF DP FLUX ALGORITHMS` — so we need to keep them secure.

We looked through a cryptography textbook for a few minutes, and we learned that it is very hard to factor products of two large prime numbers, so we devised an encryption scheme based on that fact. First, we made some preparations:

- We chose 26 different prime numbers, none of which is larger than some integer **N**.
- We sorted those primes in increasing order. Then, we assigned the smallest prime to the letter `A`, the second smallest prime to the letter `B`, and so on.
- Everyone on the team memorized this list.

Now, whenever we want to send a pangram as a message, we first remove all spacing to form a plaintext message. Then we write down the product of the prime for the first letter of the plaintext and the prime for the second letter of the plaintext. Then we write down the product of the primes for the second and third plaintext letters, and so on, ending with the product of the primes for the next-to-last and last plaintext letters. This new list of values is our ciphertext. The number of values is one smaller than the number of characters in the plaintext message.

For example, suppose that **N** = 103 and we chose to use the first 26 odd prime numbers, because we worry that it is too easy to factor even numbers. Then `A` = 3, `B` = 5, `C` = 7, `D` = 11, and so on, up to `Z` = 103. Also suppose that we want to encrypt the `CJ QUIZ`... pangram above, so our plaintext is `CJQUIZKNOWBEVYOFDPFLUXALGORITHMS`. Then the first value in our ciphertext is 7 (the prime for `C`) times 31 (the prime for `J`) = `217`; the next value is `1891`, and so on, ending with `3053`.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA
1	3	5	7	11	13	17	19	23	29	31	37	41	43	47	53	59	61	67	71	73	79	83	89	97	101	103	
2	9	15	21	33	39	51	57	69	87	93	111	123	129	141	159	177	183	201	213	219	237	249	267	291	303	309	3
3	15	25	35	55	65	85	95	115	145	155	185	205	215	235	265	295	305	335	355	365	395	415	445	485	505	515	5
4	21	35	49	77	91	119	133	161	203	217	259	287	301	329	371	413	427	469	497	511	553	581	623	679	707	721	7
5	33	55	77	121	143	187	209	253	319	341	407	451	473	517	583	649	671	737	781	803	869	913	979	1 067	1 111	1 133	11
6	39	65	91	143	169	221	247	299	377	403	481	533	559	611	689	767	793	871	923	949	1 027	1 079	1 157	1 261	1 313	1 339	13
7	51	85	119	187	221	289	323	391	493	527	629	697	731	799	901	1 003	1 037	1 139	1 207	1 241	1 343	1 411	1 513	1 649	1 717	1 751	17
8	57	95	133	209	247	323	361	437	551	589	703	779	817	893	1 007	1 121	1 159	1 273	1 349	1 387	1 501	1 577	1 691	1 843	1 919	1 957	19
9	69	115	161	253	299	391	437	529	667	713	851	943	989	1 081	1 219	1 357	1 403	1 541	1 633	1 679	1 817	1 909	2 047	2 231	2 323	2 369	23
10	87	145	203	319	377	493	551	667	841	899	1 073	1 189	1 247	1 363	1 537	1 711	1 769	1 943	2 059	2 117	2 291	2 407	2 581	2 813	2 929	2 987	29
11	93	155	217	341	403	527	589	713	899	961	1 147	1 271	1 333	1 457	1 643	1 829	1 891	2 077	2 201	2 263	2 449	2 573	2 759	3 007	3 131	3 193	31
12	111	185	259	407	481	629	703	851	1 073	1 147	1 369	1 517	1 591	1 739	1 961	2 183	2 257	2 479	2 627	2 701	2 923	3 071	3 293	3 589	3 737	3 811	37
13	123	205	287	451	533	697	779	943	1 189	1 271	1 517	1 681	1 763	1 927	2 173	2 419	2 501	2 747	2 911	2 993	3 239	3 403	3 649	3 977	4 141	4 223	41
14	129	215	301	473	559	731	817	989	1 247	1 333	1 591	1 763	1 849	2 021	2 279	2 537	2 623	2 881	3 053	3 139	3 397	3 569	3 827	4 171	4 343	4 429	43
15	141	235	329	517	611	799	893	1 081	1 363	1 457	1 739	1 927	2 021	2 209	2 491	2 773	2 867	3 149	3 337	3 431	3 713	3 901	4 183	4 559	4 747	4 841	47
16	159	265	371	583	689	901	1 007	1 219	1 537	1 643	1 961	2 173	2 279	2 491	2 809	3 127	3 233	3 551	3 763	3 869	4 187	4 399	4 717	5 141	5 353	5 459	53
17	177	295	413	649	767	1 003	1 121	1 357	1 711	1 829	2 183	2 419	2 537	2 773	3 127	3 481	3 599	3 953	4 189	4 307	4 661	4 897	5 251	5 723	5 959	6 077	59
18	183	305	427	671	793	1 037	1 159	1 403	1 769	1 891	2 257	2 501	2 623	2 867	3 233	3 599	3 721	4 087	4 331	4 453	4 819	5 063	5 429	5 917	6 161	6 283	61
19	201	335	469	737	871	1 139	1 273	1 541	1 943	2 077	2 479	2 747	2 881	3 149	3 551	3 953	4 087	4 489	4 757	4 891	5 293	5 561	5 963	6 499	6 767	6 901	67
20	213	355	497	781	923	1 207	1 349	1 633	2 059	2 201	2 627	2 911	3 053	3 337	3 763	4 189	4 331	4 757	5 041	5 183	5 609	5 893	6 319	6 887	7 171	7 313	71
21	219	365	511	803	949	1 241	1 387	1 679	2 117	2 263	2 701	2 993	3 139	3 431	3 869	4 307	4 453	4 891	5 183	5 329	5 767	6 059	6 497	7 081	7 373	7 519	73
22	237	395	553	869	1 027	1 343	1 501	1 817	2 291	2 449	2 923	3 239	3 397	3 713	4 187	4 661	4 819	5 293	5 609	5 767	6 241	6 557	7 031	7 663	7 979	8 137	79
23	249	415	581	913	1 079	1 411	1 577	1 909	2 407	2 573	3 071	3 403	3 569	3 901	4 399	4 897	5 063	5 561	5 893	6 059	6 557	6 889	7 387	8 051	8 383	8 549	83
24	267	445	623	979	1 157	1 513	1 691	2 047	2 581	2 759	3 293	3 649	3 827	4 183	4 717	5 251	5 429	5 963	6 319	6 497	7 031	7 387	7 921	8 633	8 989	9 167	89
25	291	485	679	1 067	1 261	1 649	1 843	2 231	2 813	3 007	3 589	3 977	4 171	4 559	5 141	5 723	5 917	6 499	6 887	7 081	7 663	8 051	8 633	9 409	9 797	9 991	97
26	303	505	707	1 111	1 313	1 717	1 919	2 323	2 929	3 131	3 737	4 141	4 343	4 747	5 353	5 959	6 161	6 767	7 171	7 373	7 979	8 383	8 989	9 797	10 201	10 403	101
27	309	515	721	1 133	1 339	1 751	1 957	2 369	2 987	3 193	3 811	4 223	4 429	4 841	5 459	6 077	6 283	6 901	7 313	7 519	8 137	8 549	9 167	9 991	10 403	10 609	103

We will give you a ciphertext message and the value of **N** that we used. We will not tell you which primes we used, or how to decrypt the ciphertext. Do you think you can recover the plaintext anyway?

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow; each test case consists of two lines. The first line contains two integers: **N**, as described above, and **L**, the length of the list of values in the ciphertext. The second line contains**L** integers: the list of values in the ciphertext.

Output

For each test case, output one line containing `Case #x: y`, where `x` is the test case number (starting from 1) and `y` is a string of **L** + 1 uppercase English alphabet letters: the plaintext.

Limits

1 ≤ **T** ≤ 100.
Time limit: 20 seconds per test set.
Memory limit: 1 GB.
25 ≤ **L** ≤ 100.
The plaintext contains each English alphabet letter at least once.

Test set 1 (Visible)

101 ≤ **N** ≤ 10000.

Test set 2 (Hidden)

101 ≤ **N** ≤ 10^{100}.

Sample

Input

```
2
103 31
217 1891 4819 2291 2987 3811 1739 2491 4717 445 65 1079 8383 5353 901 187 649 1003
697 3239 7663 291 123 779 1007 3551 1943 2117 1679 989 3053
10000 25
3292937 175597 18779 50429 375469 1651121 2102 3722 2376497 611683 489059 2328901
3150061 829981 421301 76409 38477 291931 730241 959821 1664197 3057407 4267589
4729181 5335543
```

Output

```
Case #1: CJQUIZKNOWBEVYOFDPFLUXALGORITHMS
Case #2: SUBDERMATOGLYPHICFJKNQVWXZ
```