

2023/24 Team Selection Test

Editorial

Felix Lo Jack Wong

3 July 2024

Statistics

	Problem	Attempts	Full	Max	Mean	Std Dev	LQ	Median	UQ	Subtasks	First Solve
TS240	Relaxing Sunday	16	20	20	16.5	6.432	16	20	20	5 14 7 14 8 12	Wong Ho Ki 00:05:01
TS241	Character Repetition	12	30	30	21.5	12.835	9	30	30	14 9 4 9 8 8 4 8	Fung Hiu Nok 00:05:18
TS242	Fuk Luk Sau	14	40	40	20.857	18.466	0	20	40	6 10 12 7 6 7 10 7 6 6	Hui Ka Wai 00:31:08
TS243	Grand School Award	7	65	65	38.429	26.928	0	37	65	12 5 20 3 15 5 10 5 8 3	Fok Kin Wing 00:58:29
TS244	Tiers of the Kingdom	4	95	24	6	10.392	0	0	12	9 1 15 1 13 0 16 0 18 0 24 0	
		16	250	179	61.5	59.699	7	47	127		

Relaxing Sunday

TS240

Problem Statement

Given

the day of the week of the first day S in a year

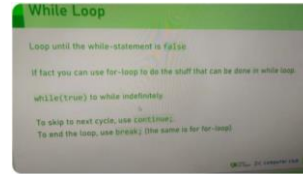


Determine

whether the d^{th} day of the year is Sunday

Subtask 1

$d = 1$



呢條友係唔係唔識英文完全唔明佢噏乜

Assume that you are not some OI Team trainers,
you should understand this if you understand English...
“If the first day of the year is Sunday, then the 1st day of the year is Sunday.
Otherwise, the 1st day of the year is not Sunday.”

Just use a single if-else statement to determine the output!
If S is **Sunday**, output **Relax**. Otherwise, output **Angry**.

Expected Score

5

Cumulative

5

Subtask 2

$S = \text{Sunday}$

The first day of the year is Sunday.

When is the next Sunday?

The 8th, 15th, 22nd, ..., 365th day!

Can you find the pattern?

When $d = 7k + 1$ where k is integer, the queried day is Sunday.

In other words, only if $d - 1$ divisible by 7, the output is Relax.

In our code, we can check this by checking whether $(d-1) \% 7$ equals to 0.

Expected Score

7

Cumulative

12

Full Solution

What about $S \neq \text{Sunday}$?

Can we modify the condition a bit?

When $S = \text{Saturday}$, $d = 7k + 2$ if it is Sunday.

When $S = \text{Friday}$, $d = 7k + 3$ if it is Sunday.

$\vdots$

$\vdots$

Basically $d = 7k + x$ if the queried day is Sunday.

Thus only if $d - x$ divisible by 7, the output is Relax.

In our code, we can check whether $(d - x) \% 7$ equals to 0.

S	x
Sunday	1
Saturday	2
Friday	3
Thursday	4
Wednesday	5
Tuesday	6
Monday	7

Expected Score 20

Cumulative 20

Common Errors

Compilation Error



```
#include <bits/stdc++.h>
using namespace std;
```

```
int main(){
    string S,d;
    cin >> S;
    if(S == "Sunday"){
        cout << "Relax" << endl;
    }else if (d = 1){
        cout << "Relax" << endl;
    }else{
        cout >> "Angry" << endl;
    }
}
```

d should be int

Too many }

Your code should pass ALL possible cases within the constraints!

Not just Sample Tests!

SAMPLE TESTS

	Input	Output
1	Monday 1	Angry
This sample satisfies the constraints of Subtask 1.		
2	Sunday 15	Relax
This sample satisfies the constraints of Subtask 2.		
3	Saturday 53	Angry
4	Wednesday 222	Relax

```
int main(){
    int d,n[100001];
    string S;
    cin>>S;
    cin>>d;
    if(S=="Sunday"&&(d-1)%7==0){
        S="Relax";
        cout<<S<<endl;
    }else if(S=="Wednesday"&&(d-1)%7==3){
        S="Relax";
        cout<<S<<endl;
    }else{
        S="Angry";
        cout<<S<<endl;
    }
    return 0;
}
```

Character Repetition

TS241

Problem Statement

Given

a string S which consists of English letters and spaces



Delete consecutive repeated characters
such that every neighbouring characters are distinct
where the comparison is case-insensitive (e.g. A equals to a)

Subtask 1

S consists of lowercase English letters (a-z) only

We don't need to handle space or uppercase letters.

Therefore to solve this problem, we can...

- Input the string by using `cin`
- Use a for-loop with `i` ranging from `0` to `S.size()-1` to check repetition

For each iteration, output `S[i]` if `S[i] != S[i-1]`

Note that you should output `S[0]` directly when `i = 0` as there is no `S[-1]`

Expected Score

14

Cumulative

14

Subtask 2

S consists of lowercase English letters (a–z) and spaces only

With spaces in S , `cin` no longer works!

We can use `getline(cin, S)` to get S .
Then just apply Subtask 1's for-loop.

Get Input with Spaces

Remember this in an earlier chapter?

Input	Chris Wong
<code>S</code>	Chris

Why not Chris Wong?
This will be discussed later.

Actually, we can use `getline(cin, S)` to get an input string S with spaces rather than using `cin >> S`.

Input	Hello, World!
<code>S</code> when using <code>cin >> S</code>	Hello,
<code>S</code> when using <code>getline(cin, S)</code>	Hello, World!

 FOUNDATION COURSE TWGSS OI Team

30

2023/24 C++ Foundation Course
Chapter 6 - Arrays

Expected Score

18

Cumulative

18

Subtask 3

S consists of English letters (A-Z, a-z) only

As the comparison is case-insensitive, we can first write a function that converts characters to lowercase before checking whether they are the same.

```
char low(char c) {  
    if (c >= 'A' && c <= 'Z')  
        c = c - 'A' + 'a';  
    return c;  
}
```

check if c is uppercase
change c to lowercase

Subtask 3

S consists of English letters (A-Z, a-z) only

Adding case-insensitive comparison to Subtask 1 solution...

- Input the string by using `cin`
- Use a for-loop with `i` ranging from `0` to `S.size()-1` to check repetition

For each iteration, output `S[i]` if `low(S[i]) != low(S[i-1])`

Note that you should output `S[0]` directly when `i = 0` as there is no `S[-1]`

Expected Score

22

Cumulative

26

Full Solution

Combining the ideas of Subtask 2 and Subtask 3...

- Using `getline()` for input
- Implement case-insensitive comparison

That's it.

Expected Score

30

Cumulative

30

Common Errors

- You cannot delete a character in a string by setting it to 0

and you should test using Sample Tests!

INPUT		OUTPUT	
1	halllloooo	1	hal...o...

```
int main(){
    string str;
    cin >> str;
    for(int i=str.length()-1; i>0; i--){
        if(str[i]==str[i-1]){
            str[i]=0;
        }
    }
    cout << str;
}
```

- You must work very hard for this masterpiece!
Sadly, your hard work didn't work!

(cannot fit in the page vertically)

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 int main()
5 {
6     string S;
7     char A[26];
8     gettime(&in,5);
9     for (int i=1;i<S.size();i++){
10         if(S[i]==S[i-1]){
11             if(S[i-1]=='A'..Z) S[i-1]='a';
12             else S[i-1]='0';
13             vec.push_back(S[i-1]);
14             S[i-1]='b';
15             vec.push_back(S[i]);
16             S[i]='c';
17             S[i-1]='c';
18             vec.push_back(S[i]);
19             S[i]='d';
20             S[i-1]='d';
21             vec.push_back(S[i]);
22             S[i]='e';
23             S[i-1]='e';
24             vec.push_back(S[i]);
25             S[i]='f';
26             S[i-1]='f';
27             vec.push_back(S[i]);
28             S[i]='g';
29             S[i-1]='g';
30             vec.push_back(S[i]);
31             S[i]='h';
32             S[i-1]='h';
33             vec.push_back(S[i]);
34             S[i]='i';
35             S[i-1]='i';
36             vec.push_back(S[i]);
37             S[i]='j';
38             S[i-1]='j';
39             vec.push_back(S[i]);
40             S[i]='k';
41             S[i-1]='k';
42             vec.push_back(S[i]);
43             S[i]='l';
44             S[i-1]='l';
45             vec.push_back(S[i]);
46             S[i]='m';
47             S[i-1]='m';
48             vec.push_back(S[i]);
49             S[i]='n';
50             S[i-1]='n';
51             vec.push_back(S[i]);
52             S[i]='o';
53             S[i-1]='o';
54             vec.push_back(S[i]);
55             S[i]='p';
56             S[i-1]='p';
57             vec.push_back(S[i]);
58             S[i]='q';
59             S[i-1]='q';
60             vec.push_back(S[i]);
61             S[i]='r';
62             S[i-1]='r';
63             vec.push_back(S[i]);
64             S[i]='s';
65             S[i-1]='s';
66             vec.push_back(S[i]);
67             S[i]='t';
68             S[i-1]='t';
69             vec.push_back(S[i]);
70             S[i]='u';
71             S[i-1]='u';
72             vec.push_back(S[i]);
73             S[i]='v';
74             S[i-1]='v';
75             vec.push_back(S[i]);
76             S[i]='w';
77             S[i-1]='w';
78             vec.push_back(S[i]);
79             S[i]='x';
80             S[i-1]='x';
81             vec.push_back(S[i]);
82             S[i]='y';
83             S[i-1]='y';
84             vec.push_back(S[i]);
85             S[i]='z';
86             S[i-1]='z';
87             vec.push_back(S[i]);
88             S[i]='A';
89             S[i-1]='A';
90             vec.push_back(S[i]);
91             S[i]='B';
92             S[i-1]='B';
93             vec.push_back(S[i]);
94             S[i]='C';
95             S[i-1]='C';
96             vec.push_back(S[i]);
97             S[i]='D';
98             S[i-1]='D';
99             vec.push_back(S[i]);
100            S[i]='E';
101            S[i-1]='E';
102            vec.push_back(S[i]);
103            S[i]='F';
104            S[i-1]='F';
105            vec.push_back(S[i]);
106            S[i]='G';
107            S[i-1]='G';
108            vec.push_back(S[i]);
109            S[i]='H';
110            S[i-1]='H';
111            vec.push_back(S[i]);
112            S[i]='I';
113            S[i-1]='I';
114            vec.push_back(S[i]);
115            S[i]='J';
116            S[i-1]='J';
117            vec.push_back(S[i]);
118            S[i]='K';
119            S[i-1]='K';
120            vec.push_back(S[i]);
121            S[i]='L';
122            S[i-1]='L';
123            vec.push_back(S[i]);
124            S[i]='M';
125            S[i-1]='M';
126            vec.push_back(S[i]);
127            S[i]='N';
128            S[i-1]='N';
129            vec.push_back(S[i]);
130            S[i]='O';
131            S[i-1]='O';
132            vec.push_back(S[i]);
133            S[i]='P';
134            S[i-1]='P';
135            vec.push_back(S[i]);
136            S[i]='Q';
137            S[i-1]='Q';
138            vec.push_back(S[i]);
139            S[i]='R';
140            S[i-1]='R';
141            vec.push_back(S[i]);
142            S[i]='S';
143            S[i-1]='S';
144            vec.push_back(S[i]);
145            S[i]='T';
146            S[i-1]='T';
147            vec.push_back(S[i]);
148            S[i]='U';
149            S[i-1]='U';
150            vec.push_back(S[i]);
151            S[i]='V';
152            S[i-1]='V';
153            vec.push_back(S[i]);
154            S[i]='W';
155            S[i-1]='W';
156            vec.push_back(S[i]);
157            S[i]='X';
158            S[i-1]='X';
159            vec.push_back(S[i]);
160            S[i]='Y';
161            S[i-1]='Y';
162            vec.push_back(S[i]);
163            S[i]='Z';
164            S[i-1]='Z';
165            vec.push_back(S[i]);
166            S[i]='a';
167            S[i-1]='a';
168            vec.push_back(S[i]);
169            S[i]='b';
170            S[i-1]='b';
171            vec.push_back(S[i]);
172            S[i]='c';
173            S[i-1]='c';
174            vec.push_back(S[i]);
175            S[i]='d';
176            S[i-1]='d';
177            vec.push_back(S[i]);
178            S[i]='e';
179            S[i-1]='e';
180            vec.push_back(S[i]);
181            S[i]='f';
182            S[i-1]='f';
183            vec.push_back(S[i]);
184            S[i]='g';
185            S[i-1]='g';
186            vec.push_back(S[i]);
187            S[i]='h';
188            S[i-1]='h';
189            vec.push_back(S[i]);
190            S[i]='i';
191            S[i-1]='i';
192            vec.push_back(S[i]);
193            S[i]='j';
194            S[i-1]='j';
195            vec.push_back(S[i]);
196            S[i]='k';
197            S[i-1]='k';
198            vec.push_back(S[i]);
199            S[i]='l';
200            S[i-1]='l';
201            vec.push_back(S[i]);
202            S[i]='m';
203            S[i-1]='m';
204            vec.push_back(S[i]);
205            S[i]='n';
206            S[i-1]='n';
207            vec.push_back(S[i]);
208            S[i]='o';
209            S[i-1]='o';
210            vec.push_back(S[i]);
211            S[i]='p';
212            S[i-1]='p';
213            vec.push_back(S[i]);
214            S[i]='q';
215            S[i-1]='q';
216            vec.push_back(S[i]);
217            S[i]='r';
218            S[i-1]='r';
219            vec.push_back(S[i]);
220            S[i]='s';
221            S[i-1]='s';
222            vec.push_back(S[i]);
223            S[i]='t';
224            S[i-1]='t';
225            vec.push_back(S[i]);
226            S[i]='u';
227            S[i-1]='u';
228            vec.push_back(S[i]);
229            S[i]='v';
230            S[i-1]='v';
231            vec.push_back(S[i]);
232            S[i]='w';
233            S[i-1]='w';
234            vec.push_back(S[i]);
235            S[i]='x';
236            S[i-1]='x';
237            vec.push_back(S[i]);
238            S[i]='y';
239            S[i-1]='y';
240            vec.push_back(S[i]);
241            S[i]='z';
242            S[i-1]='z';
243            vec.push_back(S[i]);
244            S[i]='A';
245            S[i-1]='A';
246            vec.push_back(S[i]);
247            S[i]='B';
248            S[i-1]='B';
249            vec.push_back(S[i]);
250            S[i]='C';
251            S[i-1]='C';
252            vec.push_back(S[i]);
253            S[i]='D';
254            S[i-1]='D';
255            vec.push_back(S[i]);
256            S[i]='E';
257            S[i-1]='E';
258            vec.push_back(S[i]);
259            S[i]='F';
260            S[i-1]='F';
261            vec.push_back(S[i]);
262            S[i]='G';
263            S[i-1]='G';
264            vec.push_back(S[i]);
265            S[i]='H';
266            S[i-1]='H';
267            vec.push_back(S[i]);
268            S[i]='I';
269            S[i-1]='I';
270            vec.push_back(S[i]);
271            S[i]='J';
272            S[i-1]='J';
273            vec.push_back(S[i]);
274            S[i]='K';
275            S[i-1]='K';
276            vec.push_back(S[i]);
277            S[i]='L';
278            S[i-1]='L';
279            vec.push_back(S[i]);
280            S[i]='M';
281            S[i-1]='M';
282            vec.push_back(S[i]);
283            S[i]='N';
284            S[i-1]='N';
285            vec.push_back(S[i]);
286            S[i]='O';
287            S[i-1]='O';
288            vec.push_back(S[i]);
289            S[i]='P';
290            S[i-1]='P';
291            vec.push_back(S[i]);
292            S[i]='Q';
293            S[i-1]='Q';
294            vec.push_back(S[i]);
295            S[i]='R';
296            S[i-1]='R';
297            vec.push_back(S[i]);
298            S[i]='S';
299            S[i-1]='S';
300            vec.push_back(S[i]);
301            S[i]='T';
302            S[i-1]='T';
303            vec.push_back(S[i]);
304            S[i]='U';
305            S[i-1]='U';
306            vec.push_back(S[i]);
307            S[i]='V';
308            S[i-1]='V';
309            vec.push_back(S[i]);
310            S[i]='W';
311            S[i-1]='W';
312            vec.push_back(S[i]);
313            S[i]='X';
314            S[i-1]='X';
315            vec.push_back(S[i]);
316            S[i]='Y';
317            S[i-1]='Y';
318            vec.push_back(S[i]);
319            S[i]='Z';
320            S[i-1]='Z';
321            vec.push_back(S[i]);
322            S[i]='a';
323            S[i-1]='a';
324            vec.push_back(S[i]);
325            S[i]='b';
326            S[i-1]='b';
327            vec.push_back(S[i]);
328            S[i]='c';
329            S[i-1]='c';
330            vec.push_back(S[i]);
331            S[i]='d';
332            S[i-1]='d';
333            vec.push_back(S[i]);
334            S[i]='e';
335            S[i-1]='e';
336            vec.push_back(S[i]);
337            S[i]='f';
338            S[i-1]='f';
339            vec.push_back(S[i]);
340            S[i]='g';
341            S[i-1]='g';
342            vec.push_back(S[i]);
343            S[i]='h';
344            S[i-1]='h';
345            vec.push_back(S[i]);
346            S[i]='i';
347            S[i-1]='i';
348            vec.push_back(S[i]);
349            S[i]='j';
350            S[i-1]='j';
351            vec.push_back(S[i]);
352            S[i]='k';
353            S[i-1]='k';
354            vec.push_back(S[i]);
355            S[i]='l';
356            S[i-1]='l';
357            vec.push_back(S[i]);
358            S[i]='m';
359            S[i-1]='m';
360            vec.push_back(S[i]);
361            S[i]='n';
362            S[i-1]='n';
363            vec.push_back(S[i]);
364            S[i]='o';
365            S[i-1]='o';
366            vec.push_back(S[i]);
367            S[i]='p';
368            S[i-1]='p';
369            vec.push_back(S[i]);
370            S[i]='q';
371            S[i-1]='q';
372            vec.push_back(S[i]);
373            S[i]='r';
374            S[i-1]='r';
375            vec.push_back(S[i]);
376            S[i]='s';
377            S[i-1]='s';
378            vec.push_back(S[i]);
379            S[i]='t';
380            S[i-1]='t';
381            vec.push_back(S[i]);
382            S[i]='u';
383            S[i-1]='u';
384            vec.push_back(S[i]);
385            S[i]='v';
386            S[i-1]='v';
387            vec.push_back(S[i]);
388            S[i]='w';
389            S[i-1]='w';
390            vec.push_back(S[i]);
391            S[i]='x';
392            S[i-1]='x';
393            vec.push_back(S[i]);
394            S[i]='y';
395            S[i-1]='y';
396            vec.push_back(S[i]);
397            S[i]='z';
398            S[i-1]='z';
399            vec.push_back(S[i]);
400            S[i]='A';
401            S[i-1]='A';
402            vec.push_back(S[i]);
403            S[i]='B';
404            S[i-1]='B';
405            vec.push_back(S[i]);
406            S[i]='C';
407            S[i-1]='C';
408            vec.push_back(S[i]);
409            S[i]='D';
410            S[i-1]='D';
411            vec.push_back(S[i]);
412            S[i]='E';
413            S[i-1]='E';
414            vec.push_back(S[i]);
415            S[i]='F';
416            S[i-1]='F';
417            vec.push_back(S[i]);
418            S[i]='G';
419            S[i-1]='G';
420            vec.push_back(S[i]);
421            S[i]='H';
422            S[i-1]='H';
423            vec.push_back(S[i]);
424            S[i]='I';
425            S[i-1]='I';
426            vec.push_back(S[i]);
427            S[i]='J';
428            S[i-1]='J';
429            vec.push_back(S[i]);
430            S[i]='K';
431            S[i-1]='K';
432            vec.push_back(S[i]);
433            S[i]='L';
434            S[i-1]='L';
435            vec.push_back(S[i]);
436            S[i]='M';
437            S[i-1]='M';
438            vec.push_back(S[i]);
439            S[i]='N';
440            S[i-1]='N';
441            vec.push_back(S[i]);
442            S[i]='O';
443            S[i-1]='O';
444            vec.push_back(S[i]);
445            S[i]='P';
446            S[i-1]='P';
447            vec.push_back(S[i]);
448            S[i]='Q';
449            S[i-1]='Q';
450            vec.push_back(S[i]);
451            S[i]='R';
452            S[i-1]='R';
453            vec.push_back(S[i]);
454            S[i]='S';
455            S[i-1]='S';
456            vec.push_back(S[i]);
457            S[i]='T';
458            S[i-1]='T';
459            vec.push_back(S[i]);
460            S[i]='U';
461            S[i-1]='U';
462            vec.push_back(S[i]);
463            S[i]='V';
464            S[i-1]='V';
465            vec.push_back(S[i]);
466            S[i]='W';
467            S[i-1]='W';
468            vec.push_back(S[i]);
469            S[i]='X';
470            S[i-1]='X';
471            vec.push_back(S[i]);
472            S[i]='Y';
473            S[i-1]='Y';
474            vec.push_back(S[i]);
475            S[i]='Z';
476            S[i-1]='Z';
477            vec.push_back(S[i]);
478            S[i]='a';
479            S[i-1]='a';
480            vec.push_back(S[i]);
481            S[i]='b';
482            S[i-1]='b';
483            vec.push_back(S[i]);
484            S[i]='c';
485            S[i-1]='c';
486            vec.push_back(S[i]);
487            S[i]='d';
488            S[i-1]='d';
489            vec.push_back(S[i]);
490            S[i]='e';
491            S[i-1]='e';
492            vec.push_back(S[i]);
493            S[i]='f';
494            S[i-1]='f';
495            vec.push_back(S[i]);
496            S[i]='g';
497            S[i-1]='g';
498            vec.push_back(S[i]);
499            S[i]='h';
500            S[i-1]='h';
501            vec.push_back(S[i]);
502            S[i]='i';
503            S[i-1]='i';
504            vec.push_back(S[i]);
505            S[i]='j';
506            S[i-1]='j';
507            vec.push_back(S[i]);
508            S[i]='k';
509            S[i-1]='k';
510            vec.push_back(S[i]);
511            S[i]='l';
512            S[i-1]='l';
513            vec.push_back(S[i]);
514            S[i]='m';
515            S[i-1]='m';
516            vec.push_back(S[i]);
517            S[i]='n';
518            S[i-1]='n';
519            vec.push_back(S[i]);
520            S[i]='o';
521            S[i-1]='o';
522            vec.push_back(S[i]);
523            S[i]='p';
524            S[i-1]='p';
525            vec.push_back(S[i]);
526            S[i]='q';
527            S[i-1]='q';
528            vec.push_back(S[i]);
529            S[i]='r';
530            S[i-1]='r';
531            vec.push_back(S[i]);
532            S[i]='s';
533            S[i-1]='s';
534            vec.push_back(S[i]);
535            S[i]='t';
536            S[i-1]='t';
537            vec.push_back(S[i]);
538            S[i]='u';
539            S[i-1]='u';
540            vec.push_back(S[i]);
541            S[i]='v';
542            S[i-1]='v';
543            vec.push_back(S[i]);
544            S[i]='w';
545            S[i-1]='w';
546            vec.push_back(S[i]);
547            S[i]='x';
548            S[i-1]='x';
549            vec.push_back(S[i]);
550            S[i]='y';
551            S[i-1]='y';
552            vec.push_back(S[i]);
553            S[i]='z';
554            S[i-1]='z';
555            vec.push_back(S[i]);
556            S[i]='A';
557            S[i-1]='A';
558            vec.push_back(S[i]);
559            S[i]='B';
560            S[i-1]='B';
561            vec.push_back(S[i]);
562            S[i]='C';
563            S[i-1]='C';
564            vec.push_back(S[i]);
565            S[i]='D';
566            S[i-1]='D';
567            vec.push_back(S[i]);
568            S[i]='E';
569            S[i-1]='E';
570            vec.push_back(S[i]);
571            S[i]='F';
572            S[i-1]='F';
573            vec.push_back(S[i]);
574            S[i]='G';
575            S[i-1]='G';
576            vec.push_back(S[i]);
577            S[i]='H';
578            S[i-1]='H';
579            vec.push_back(S[i]);
580            S[i]='I';
581            S[i-1]='I';
582            vec.push_back(S[i]);
583            S[i]='J';
584            S[i-1]='J';
585            vec.push_back(S[i]);
586            S[i]='K';
587            S[i-1]='K';
588            vec.push_back(S[i]);
589            S[i]='L';
590            S[i-1]='L';
591            vec.push_back(S[i]);
592            S[i]='M';
593            S[i-1]='M';
594            vec.push_back(S[i]);
595            S[i]='N';
596            S[i-1]='N';
597            vec.push_back(S[i]);
598            S[i]='O';
599            S[i-1]='O';
600            vec.push_back(S[i]);
601            S[i]='P';
602            S[i-1]='P';
603            vec.push_back(S[i]);
604            S[i]='Q';
605            S[i-1]='Q';
606            vec.push_back(S[i]);
607            S[i]='R';
608            S[i-1]='R';
609            vec.push_back(S[i]);
610            S[i]='S';
611            S[i-1]='S';
612            vec.push_back(S[i]);
613            S[i]='T';
614            S[i-1]='T';
615            vec.push_back(S[i]);
616            S[i]='U';
617            S[i-1]='U';
618            vec.push_back(S[i]);
619            S[i]='V';
620            S[i-1]='V';
621            vec.push_back(S[i]);
622            S[i]='W';
623            S[i-1]='W';
624            vec.push_back(S[i]);
625            S[i]='X';
626            S[i-1]='X';
627            vec.push_back(S[i]);
628            S[i]='Y';
629            S[i-1]='Y';
630            vec.push_back(S[i]);
631            S[i]='Z';
632            S[i-1]='Z';
633            vec.push_back(S[i]);
634            S[i]='a';
635            S[i-1]='a';
636            vec.push_back(S[i]);
637            S[i]='b';
638            S[i-1]='b';
639            vec.push_back(S[i]);
640            S[i]='c';
641            S[i-1]='c';
642            vec.push_back(S[i]);
643            S[i]='d';
644            S[i-1]='d';
645            vec.push_back(S[i]);
646            S[i]='e';
647            S[i-1]='e';
648            vec.push_back(S[i]);
649            S[i]='f';
650            S[i-1]='f';
651            vec.push_back(S[i]);
652            S[i]='g';
653            S[i-1]='g';
654            vec.push_back(S[i]);
655            S[i]='h';
656            S[i-1]='h';
657            vec.push_back(S[i]);
658            S[i]='i';
659            S[i-1]='i';
660            vec.push_back(S[i]);
661            S[i]='j';
662            S[i-1]='j';
663            vec.push_back(S[i]);
664            S[i]='k';
665            S[i-1]='k';
666            vec.push_back(S[i]);
667            S[i]='l';
668            S[i-1]='l';
669            vec.push_back(S[i]);
670            S[i]='m';
671            S[i-1]='m';
672            vec.push_back(S[i]);
673            S[i]='n';
674            S[i-1]='n';
675            vec.push_back(S[i]);
676            S[i]='o';
677            S[i-1]='o';
678            vec.push_back(S[i]);
679            S[i]='p';
680            S[i-1]='p';
681            vec.push_back(S[i]);
682            S[i]='q';
683            S[i-1]='q';
684            vec.push_back(S[i]);
685            S[i]='r';
686            S[i-1]='r';
687            vec.push_back(S[i]);
688            S[i]='s';
689            S[i-1]='s';
690            vec.push_back(S[i]);
691            S[i]='t';
692            S[i-1]='t';
693            vec.push_back(S[i]);
694            S[i]='u';
695            S[i-1]='u';
696            vec.push_back(S[i]);
697            S[i]='v';
698            S[i-1]='v';
699            vec.push_back(S[i]);
700            S[i]='w';
701            S[i-1]='w';
702            vec.push_back(S[i]);
703            S[i]='x';
704            S[i-1]='x';
705            vec.push_back(S[i]);
706            S[i]='y';
707            S[i-1]='y';
708            vec.push_back(S[i]);
709            S[i]='z';
710            S[i-1]='z';
711            vec.push_back(S[i]);
712            S[i]='A';
713            S[i-1]='A';
714            vec.push_back(S[i]);
715            S[i]='B';
716            S[i-1]='B';
717            vec.push_back(S[i]);
718            S[i]='C';
719            S[i-1]='C';
720            vec.push_back(S[i]);
721            S[i]='D';
722            S[i-1]='D';
723            vec.push_back(S[i]);
724            S[i]='E';
725            S[i-1]='E';
726            vec.push_back(S[i]);
727            S[i]='F';
728            S[i-1]='F';
729            vec.push_back(S[i]);
730            S[i]='G';
731            S[i-1]='G';
732            vec.push_back(S[i]);
733            S[i]='H';
734            S[i-1]='H';
735            vec.push_back(S[i]);
736            S[i]='I';
737            S[i-1]='I';
738            vec.push_back(S[i]);
739            S[i]='J';
740            S[i-1]='J';
741            vec.push_back(S[i]);
742            S[i]='K';
743            S[i-1]='K';
744            vec.push_back(S[i]);
745            S[i]='L';
746            S[i-1]='L';
747            vec.push_back(S[i]);
748            S[i]='M';
749            S[i-1]='M';
750            vec.push_back(S[i]);
751            S[i]='N';
752            S[i-1]='N';
753            vec.push_back(S[i]);
754            S[i]='O';
755            S[i-1]='O';
756            vec.push_back(S[i]);
757            S[i]='P';
758            S[i-1]='P';
759            vec.push_back(S[i]);
760            S[i]='Q';
761            S[i-1]='Q';
762            vec.push_back(S[i]);
763            S[i]='R';
764            S[i-1]='R';
765            vec.push_back(S[i]);
766            S[i]='S';
767            S[i-1]='S';
768            vec.push_back(S[i]);
769            S[i]='T';
770            S[i-1]='T';
771            vec.push_back(S[i]);
772            S[i]='U';
773            S[i-1]='U';
774            vec.push_back(S[i]);
775            S[i]='V';
776            S[i-1]='V';
777            vec.push_back(S[i]);
778            S[i]='W';
779            S[i-1]='W';
780            vec.push_back(S[i]);
781            S[i]='X';
782            S[i-1]='X';
783            vec.push_back(S[i]);
784            S[i]='Y';
785            S[i-1]='Y';
786            vec.push_back(S[i]);
787            S[i]='Z';
788            S[i-1]='Z';
789            vec.push_back(S[i]);
790            S[i]='a';
791            S[i-1]='a';
792            vec.push_back(S[i]);
793            S[i]='b';
794            S[i-1]='b';
795            vec.push_back(S[i]);
796            S[i]='c';
797            S[i-1]='c';
798            vec.push_back(S[i]);
799            S[i]='d';
800            S[i-1]='d';
801            vec.push_back(S[i]);
802            S[i]='e';
803            S[i-1]='e';
804            vec.push_back(S[i]);
805            S[i]='f';
806            S[i-1]='f';
807            vec.push_back(S[i]);
808            S[i]='g';
809            S[i-1]='g';
810            vec.push_back(S[i]);
811            S[i]='h';
812            S[i-1]='h';
813            vec.push_back(S[i]);
814            S[i]='i';
815            S[i-1]='i';
816            vec.push_back(S[i]);
817            S[i]='j';
818            S[i-1]='j';
819            vec.push_back(S[i]);
820            S[i]='k';
821            S[i-1]='k';
822            vec.push_back(S[i]);
823            S[i]='l';
824            S[i-1]='l';
825            vec.push_back(S[i]);
826            S[i]='m';
827            S[i-1]='m';
828            vec.push_back(S[i]);
829            S[i]='n';
830            S[i-1]='n';
831            vec.push_back(S[i]);
832            S[i]='o';
833            S[i-1]='o';
834            vec.push_back(S[i]);
835            S[i]='p';
836            S[i-1]='p';
837            vec.push_back(S[i]);
838            S[i]='q';
839            S[i-1]='q';
840            vec.push_back(S[i]);
841            S[i]='r';
842            S[i-1]='r';
843            vec.push_back(S[i]);
844            S[i]='s';
845            S[i-1]='s';
846            vec.push_back(S[i]);
847            S[i]='t';
848            S[i-1]='t';
849            vec.push_back(S[i]);
850            S[i]='u';
851            S[i-1]='u';
852            vec.push_back(S[i]);
853            S[i]='v';
854            S[i-1]='v';
855            vec.push_back(S[i]);
856            S[i]='w';
857            S[i-1]='w';
858            vec.push_back(S[i]);
859            S[i]='x';
860            S[i-1]='x';
861            vec.push_back(S[i]);
862            S[i]='y';
863            S[i-1]='y';
864            vec.push_back(S[i]);
865            S[i]='z';
866            S[i-1]='z';
867            vec.push_back(S[i]);
868            S[i]='A';
869            S[i-1]='A';
870            vec.push_back(S[i]);
871            S[i]='B';
872            S[i-1]='B';
873            vec.push_back(S[i]);
874            S[i]='C';
875            S[i-1]='C';
876            vec.push_back(S[i]);
877            S[i]='D';
878            S[i-1]='D';
879            vec.push_back(S[i]);
880            S[i]='E';
881            S[i-1]='E';
882            vec.push_back(S[i]);
883            S[i]='F';
884            S[i-1]='F';
885            vec.push_back(S[i]);
886            S[i]='G';
887            S[i-1]='G';
888            vec.push_back(S[i]);
889            S[i]='H';
890            S[i-1]='H';
891            vec.push_back(S[i]);
892            S[i]='I';
893            S[i-1]='I';
894            vec.push_back(S[i]);
895            S[i]='J';
896            S[i-1]='J';
897            vec.push_back(S[i]);
898            S[i]='K';
899            S[i-1]='K';
900            vec.push_back(S[i]);
901            S[i]='L';
902            S[i-1]='L';
903            vec.push_back(S[i]);
904            S[i]='M';
905            S[i-1]='M';
906            vec.push_back(S[i]);
907            S[i]='N';
908            S[i-1]='N';
909            vec.push_back(S[i]);
910            S[i]='O';
911            S[i-1]='O';
912            vec.push_back(S[i]);
913            S[i]='P';
914            S[i-1]='P';
915            vec.push_back(S[i]);
916            S[i]='Q';
917            S[i-1]='Q';
918            vec.push_back(S[i]);
919            S[i]='R';
920            S[i-1]='R';
921            vec.push_back(S[i]);
922            S[i]='S';
923            S[i-1]='S';
924            vec.push_back(S[i]);
925            S[i]='T';
926            S[i-1]='T';
927            vec.push_back(S[i]);
928            S[i]='U';
929            S[i-1]='U';
930            vec.push_back(S[i]);
931            S[i]='V';
932            S[i-1]='V';
933            vec.push_back(S[i]);
934            S[i]='W';
935            S[i-1]='W';
936            vec.push_back(S[i]);
937            S[i]='X';
938            S[i-1]='X';
939            vec.push_back(S[i]);
940            S[i]='Y';
941            S[i-1]='Y';
942            vec.push_back(S[i]);
943            S[i]='Z';
944            S[i-1]='Z';
945            vec.push_back(S[i]);
946            S[i]='a';
947            S[i-1]='a';
948            vec.push_back(S[i]);
949            S[i]='b';
950            S[i-1]='b';
951            vec.push_back(S[i]);
952            S[i]='c';
953            S[i-1]='c';
954            vec.push_back(S[i]);
955            S[i]='d';
956            S[i-1]='d';
957            vec.push_back(S[i]);
958            S[i]='e';
959            S[i-1]='e';
960            vec.push_back(S[i]);
961            S[i]='f';
962            S[i-1]='f';
963            vec.push_back(S[i]);
964            S[i]='g';
965            S[i-1]='g';
966            vec.push_back(S[i]);
967            S[i]='h';
968            S[i-1]='h';
969            vec.push_back(S[i]);
970            S[i]='i';
971            S[i-1]='i';
972            vec.push_back(S[i]);
973            S[i]='j';
974            S[i-1]='j';
975            vec.push_back(S[i]);
976            S[i]='k';
977            S[i-1]='k';
978            vec.push_back(S[i]);
979            S[i]='l';
980            S[i-1]='l';
981            vec.push_back(S[i]);
982            S[i]='m';
983            S[i-1]='m';
984            vec.push_back(S[i]);
985            S[i]='n';
986            S[i-1]='n';
987            vec.push_back(S[i]);
988            S[i]='o';
989            S[i-1]='o';
990            vec.push_back(S[i]);
991            S[i]='p';
992            S[i-1]='p';
993            vec.push_back(S[i]);
994            S[i]='q';
995            S[i-1]='q';
996            vec.push_back(S[i]);
997            S[i]='r';
998            S[i-1]='r';
999            vec.push_back(S[i]);
1000           S[i]='s';
1001           S[i-1]='s';
1002           vec.push_back(S[i]);
1003           S[i]='t';
1004           S[i-1]='t';
1005           vec.push_back(S[i]);
1006           S[i]='u';
1007           S[i-1]='u';
1008           vec.push_back(S[i]);
1009           S[i]='v';
1010           S[i-1]='v';
1011           vec.push_back(S[i]);
1012           S[i]='w';
1013           S[i-1]='w';
1014           vec.push_back(S[i]);
1015           S[i]='x';
1016           S[i-1]='x';
1017           vec.push_back(S[i]);
1018           S[i]='y';
1019           S[i-1]='y';
1020           vec.push_back(S[i]);
1021           S[i]='z';
1022           S[i-1]='z';
1023           vec.push_back(S[i]);
1024           S[i]='A';
1025           S[i-1]='A';
1026           vec.push_back(S[i]);
1027           S[i]='B';
1028           S[i-1]='B';
1029           vec.push_back(S[i]);
1030           S[i]='C';
1031           S[i-1]='C';
1032           vec.push_back(S[i]);
1033           S[i]='D';
1034           S[i-1]='D';
1035           vec.push_back(S[i]);
1036           S[i]='E';
1037           S[i-1]='E';
1038           vec.push_back(S[i]);
1039           S[i]='F';
1040           S[i-1]='F';
1041           vec.push_back(S[i]);
1042           S[i]='G';
1043           S[i-1]='G';
1044           vec.push_back(S[i]);
1045           S[i]='H';
1046           S[i-1]='H';
1047           vec.push_back(S[i]);
1048           S[i]='I';
1049           S[i-1]='I';
1050           vec.push_back(S[i]);
1051           S[i]='J';
1052           S[i-1]='J';
1053           vec.push_back(S[i]);
1054           S[i]='K';
1055           S[i-1]='K';
1056           vec.push_back(S[i]);
1057           S[i]='L';
1058           S[i-1]='L';
1059           vec.push_back(S[i]);
1060           S[i]='M';
1061           S[i-1]='M';
1062           vec.push_back(S[i]);
1063           S[i]='N';
1064           S[i-1]='N';
1065           vec.push_back(S[i]);
1066           S[i]='O';
1067           S[i-1]='O';
1068           vec.push_back(S[i]);
1069           S[i]='P';
1070           S[i-1]='P';
1071           vec.push_back(S[i]);
1072           S[i]='Q';
1073           S[i-1]='Q';
1074           vec.push_back(S[i]);
1075           S[i]='R';
1076           S[i-1]='R';
1077           vec.push_back(S[i]);
1078           S[i]='S';
1079           S[i-1]='S';
1080           vec.push_back(S[i]);
1081           S[i
```

Fuk Luk Sau

TS242

Problem Statement

N_F Fuk items

Package Price

P_1 P_2 $\dots$ F_{N_F}

Original Price

F_1 F_2 $\dots$ F_{N_F}

N_L Luk items

Original Price

L_1 L_2 $\dots$ L_{N_L}

N_S Sau items

Original Price

S_1 S_2 $\dots$ S_{N_S}

Answer is $\max((F_x + L_y + S_z) - P_x, 0)$

Subtask 1

$$N_F = N_L = N_S = 1$$

$$1 \leq P_i \quad F_i \quad L_i \quad S_i \leq 5000$$

From the problem statement, we know that

$$\text{answer is } \max((F_x + L_y + S_z) - P_x, 0)$$

In this subtask, $x = y = z = 1$

By substituting into the formula, answer is just $\max((F_1 + L_1 + S_1) - P_1, 0)$

Expected Score

6

Cumulative

6

Subtask 2

$$1 \leq N_F \ N_L \ N_S \leq 20$$

$$1 \leq P_i \ F_i \ L_i \ S_i \leq 5000$$

We can just brute force by trying all combinations!

- Create variable $ans = 0$
- Use three for-loops with $1 \leq x \leq N_F, 1 \leq y \leq N_L, 1 \leq z \leq N_S$
For each iteration, update $ans = \max((F_x + L_y + S_z) - P_x, ans)$

$O(N^3)$

Expected Score

18

Cumulative

18

Subtask 3

$$1 \leq N_F \quad N_L \quad N_S \leq 2000$$

$$1 \leq P_i \quad F_i \quad L_i \quad S_i \leq 5000$$

The package price actually depends on the **Fuk item** chosen!

So the selection of **Luk item** and **Sau item** can be performed separately!

Now we can rewrite the answer formula to $\max(F_x + \max(L_y + S_z) - P_x, 0)$

- Use two for-loops with $1 \leq y \leq N_L, 1 \leq z \leq N_S$ to find $\max(L_y + S_z)$
- Use a for loop with $1 \leq x \leq N_F$ to find $\max(F_x + \max(L_y + S_z) - P_x, 0)$

$O(N^2)$

Expected Score

24

Cumulative

24

Subtask 4

$$1 \leq P_i \ F_i \ L_i \ S_i \leq 5000$$

We can consider the selection of **Luk item** and **Sau item** independently!

Rewrite the answer formula to $\max(\max(F_x - P_x) + \max(L_y) + \max(S_z), 0)$

- Just calculate each part separately by each using a for-loop
- Sum up $\max(F_x - P_x)$, $\max(L_y)$, $\max(S_z)$ and compare the result with 0

0(N)

Expected Score

34

Cumulative

34

Full Solution

As $1 \leq P_i \ F_i \ L_i \ S_i \leq 10^9$, answer can be as large as 3×10^9 !

Therefore may I introduce our old friend – `long long`!

Just implement Subtask 4 solution with `long long`, done!

`O( N )`

Expected Score

40

Cumulative

40

Common Errors

- long long classics!

→ lose 6 marks (which is already very lenient!)

```
int F, L, S, Pac[200000], Fuk[200000], Luk[200000], Sau[200000], ans=0;
```

- Didn't output 0 when $(F_x + L_y + S_z) - P_x$ is negative

→ In order to remind you to output 0, you'll get 0 marks!

```
int main(){
    int a,b,c,d,f,l,s,t;
    cin>>a>>b>>c;
    cin>>d>>f>>l>>s>>t;
    t=f*a+l*b+s*c-d;
    cout<<t<<endl;
    return 0;
}
```

```
int main(){
    int a=1 ,b=1,c=1 ,d,e,f,g,h;
    cin >> a >> b >> c;
    cin >> d;
    cin >> e;
    cin >> f;
    cin >> g;
    cout << e + f + g -d << endl;
}
```

Grand School Award

TS243

Problem Statement

N data entries

For the i^{th} entry, the score P_i of the school S_i in the year Y_i is given



Ranking the schools for every year



Q queries

For the j^{th} query, output the rank of the school s_j in the year y_j

Subtask 1

$$1 \leq N, P_i \leq 5000$$

$$Q = 1$$

$$S_x = s_1 \text{ for exactly one } x$$

$$Y_1 = Y_2 = \dots = Y_N = y_1$$

Only one year with one guaranteed valid query!

- Use any data structure that can store pairs of integer (P) and string (S)
e.g. `pair<int, string> A[]` or `vector<pair<int, string>> B`
- Sort the data structure (note that `pair` will sort by `.first`)
- Loop through the data structure to find the position of the queried entry

Subtask 1

$$1 \leq N, P_i \leq 5000$$

$$Q = 1$$

$$S_x = s_1 \text{ for exactly one } x$$

$$Y_1 = Y_2 = \dots = Y_N = y_1$$

Why my program outputs 2 when 5 should be outputted in Sample Test 1?

Remember that C++ `sort()` is default to sort in ascending order, which will make the rank to be ordered in reverse! Either way can solve this problem:

- `reverse()` the data structure after sorting
- write our own compare function which sorts in descending order and put it in the `sort()` function

`O( N lgN )`

Expected Score

12

Cumulative

12

Subtask 2

$$Y_1 = Y_2 = \dots = Y_N = y_1 = y_2 = \dots = y_Q$$

Only one year!

Similar to Subtask 1, we can just...

- Use any data structure that can store pairs of integer (P) and string (S)
e.g. `pair<int, string> A[]` or `vector<pair<int, string>> A`
- Sort the data structure (note that `pair` will sort by `.first`)

Then we can just...

- Loop through the data structure to find the position of the queries
or can we...?

Subtask 2

$$Y_1 = Y_2 = \dots = Y_N = y_1 = y_2 = \dots = y_Q$$

We cannot “loop through the data structure” for every queries as this is a $O(NQ)$ move!

Instead, we can pre-compute things!

- Use `map<string, int>` for storing S and their respective rank
- Loop through the data structure once for storing the answers in the `map`
- Just take the answers from the `map` for each query

Remember to output -1 when the query is not found in the `map`!

e.g. for a `map` named `C`, `!C.count(x)` or `C.find(x) == C.end()` will returns `true` if `x` is not found in it

$O(N \lg N + Q)$

Expected Score

32

Cumulative

32

Subtask 3

$$1 \leq N, Q, Y_i, P_i, y_j \leq 5000$$

Limiting number of data and queries, and the year is bounded to 5000.

We can easily extend Subtask 1 solution for this subtask.

- Using an array of the original data structure
e.g. `pair<int, string> A[][]` or `vector<pair<int, string>> B[]`
- Sort every data structure
e.g. for $1 \leq k \leq 5000$, `sort(A[k], A[k]+N)` or `sort(B[k].begin(), B[k].end())`
- For each query, loop through the data structure of the respective year to find the position of the queried entry and output -1 when not found
e.g. loop through `A[yj]` or `B[yj]` for the j^{th} query

$O(N \lg N Q)$

Expected Score

15

Cumulative

47

Subtask 4

$$1 \leq N, Q \leq 5000$$

The year is not bounded to 5000, how can we tackle weirdly large years?

Oh! Here comes `map` again!

We can simply implement it by editing a bit of the Subtask 3 solution:

- Using ~~an array~~ a map of the original data structure
e.g. ~~`map<int, pair<int, string>> A[]`~~ (not working anymore!) or
`map<int, vector<pair<int, string>>> B`
- Sort every data structure
Remember how to loop through every element in a `map`
- For each query, the handling remains unchanged

`O( N lgN Q )`

Expected Score

37

Cumulative

57

Full Solution

In Subtask 2, we implement pre-computing to prevent $O(NQ)$.

In Subtask 3 and 4, we implement the solution for multiple years.

For the full solution, we can combine both progress into a definitive code!

- For the pre-computation, we can use `map<int, map<string, int>>` for storing different years' S and their respective rank

The remaining is just pure simple implementation!

Alternatively, you can use `struct` instead of the complicated data structures. That will keep your code cleaner and more readable!

```
struct sch{
    int yr, sc;
    string name;
}a[200005], tmp;
```

$O(N \lg N + Q)$

Expected Score

65

Cumulative

65

Tiers of the Kingdom

TS244

Problem Statement

N tiers of floating islands

Tier i islands all have power level P_i



You can merge K tier i islands to one tier $i + 1$ island

Find maximum total power level by merging islands optimally

Observation 1

Merge K tier i islands to a tier $i + 1$ islands changes the total power level by $P_{i+1} - KP_i$.



If merging islands makes a positive change, we would want to merge as much islands as we can.

Subtask 1

$$N = 2$$

There are only two possible situations:

- Merging $\left\lfloor \frac{A_1}{K} \right\rfloor K$ tier 1 islands to $\left\lfloor \frac{A_1}{K} \right\rfloor$ tier 2 islands
- No merge at all

We just have to check which of the two possible situations gives a higher total power level and output its value.

Expected Score

9

Cumulative

9

Subtask 2

$$N = 3$$

Can we do this greedily by merging tier i islands to tier $i + 1$ islands only when it will return an instant profit ($P_{i+1} > KP_i$)?

No!

Note that we may encounter cases like this:

	$K = 5$		
i	1	2	3
P_i	1	3	100
A_i	15	12	0
Correct output			300

Merging tier 1 islands to tier 2 islands seems stupid at first glance.

But if we merge tier 1 to tier 2 first, we have a total of 15 tier 2 islands, which can be further merged into 3 tier 3 islands.

Subtask 2

$$N = 3$$

Therefore, we have to consider all four situations:

- Merging tier 1 islands to tier 2 islands, then merging tier 2 islands to tier 3 islands
- Merging tier 1 islands to tier 2 islands only
- Merging tier 2 islands to tier 3 islands only
- No merge at all

Expected Score

15

Cumulative

24

Subtask 3

$$A_1 = K$$

$$0 \leq A_i < K \text{ for } 1 < i \leq N$$

There is only one merge available initially.

After merging tier 1 islands to a tier 2 island,
then if $A_2 = K - 1$, we can choose to merge again to a tier 3 island,
then if $A_3 = K - 1$, we can choose to merge again to a tier 4 island...

Basically for any $1 < i < N$, if all $A_j = K - 1$ where $1 < j \leq i$,
we can choose to merge all islands from tier 1 to i to a tier $i + 1$ island.

Subtask 3

$$A_1 = K$$

$$0 \leq A_i < K \text{ for } 1 < i \leq N$$

- Merge islands from low tier to high tier whenever possible
- Calculate the current total power level for each merge
- Keep track of the maximum total power level ever appeared

Optimally, we can make use of partial sum or other method to skip the calculation each merge, speeding up the code significantly.

$O(N^2 \text{ or } N)$

Expected Score

13

Cumulative

37

Subtask 5

$$0 \leq A_i \leq K$$

We can split the array into several parts such that each array fulfil the constraints of Subtask 3.

We can solve each array using Subtask 3 method and add the answers together.

Remember that after splitting the array, you can still merge N islands at the end of each array to the first island of the next array.

$O(N^2 \text{ or } N)$

Expected Score

31

Cumulative

55

Observation 2

There are two types of merging:

- Merging K^x tier i islands to a tier $i + x$ island
- Merging all tier $i + 1$ to tier $i + x - 1$ islands and some tier i islands to a tier $i + x$ islands

The first type of merge is worth if and only if the second type of merge is also worth.

Subtask 4

$$1 \leq N \leq 10$$

As N is small, we can loop try all possible ways to merge the islands and check whether they are worth.

Note that you would like to merge tier $i + 1$ island first as tier i islands can be merged to tier $i + 1$ island but tier $i + 1$ islands cannot be split.

$O(N^3)$

Expected Score

40

Cumulative

71

Full Solution

There are N^2 ways to merge.

Therefore we have to calculate the cost in $O(1)$ to prevent getting TLE.

- $C_{i\dots i+n} = M_{i\dots i+n} \times P_i + A_{i+1} \times P_{i+1} + \dots + A_{i+n-1} \times P_{i+n-1}$
- $C_{i\dots i+n+1} = M_{i\dots i+n+1} \times P_i + A_{i+1} \times P_{i+1} + \dots + A_{i+n} \times P_{i+n}$
 $= C_{i\dots i+n} + A_{i+n} \times P_{i+n} + (M_{i\dots i+n+1} - M_{i\dots i+n}) \times P_i$

where $M_{i\dots i+n} + A_{i+1} \times K + \dots + A_{i+n-1} \times K^{n-1} = K^n$

If you memorise $C_{i\dots i+n}$, the cost can be calculate in $O(1)$!

$O(N^2)$

Expected Score

95

Cumulative

95