Hackathon Rule Set

- No Internet Access apart allowed from access to the Submission Portal (https://hackathon-submission-portal.firebaseapp.com/C9dkPv8zhsrUR3oFzWy2)
- No use of external libraries (libraries not included in the standard library of your choosing)
- No use of pre-written code except for the skeleton files emailed out
- No using mobile phones for the purpose of receiving external help
    - Phone calls are permitted but must be made outside of the lab so as not to disturb others
- No access to H:/ Drive
- No flash storage

- Solutions must be submitted as P1, P2, P3 etc.
    - Eg. Problem 1 written in Java would be submitted as P1.java
- You are allowed to submit each problem in a different language, eg. P1 in Java, P2 in Python
- Runtime of solutions must not exceed 2 minutes otherwise automatic fail

Be excellent to each other

P0. Practice Problem

Problem Description.

Before you start participating in the Hackathon let's use a simple problem for you to become familiar with the way you must code your programs for submission.

Your task is to...

Given a file `f` containing `n` integer numbers (one number per line), read the numbers `a`, `b` and `c` from the first three lines of `f` and print their sum; if `f` contains less than three numbers, print `ERROR`).

Sample Input file: input_1.txt

```
5
3
6
4
8
9
2
```

Output to be printed:

```
14
```

Sample Input file: input_2.txt

```
3
2
```

Output to be printed:

```
ERROR
```

P1. First Date: Which Movie Do We Watch?

Problem Description.

A long time ago in a galaxy far, far away....
Saint Valentine's day has finally arrived to Endor and its forest moon! All the Ewoks are both happy and nervous to celebrate it, especially Latte and Mocha, a young couple who met in a cafe a few days ago and have started dating since.

Latte has invited Mocha to the cinema for Saint Valentine's day. To pick which movie to go to, they have agreed on looking at the Ewok Cinema Facebook Group, selecting the 3rd movie with most likes.

Your task is to...
Help Latte and Mocca to compute the number of likes for the movie with 3rd most likes.

More precisely, you will read the *likes* of a set of movies from a file `f` (each line of the file contains a single integer representing the likes of a movie), and you will have to print the number of likes for the movie with 3rd most likes (if the list does not have at least 3 movies then print ERROR).

Sample Input file: input_1.txt

```
5
3
6
4
8
9
2
```

Output to be printed:

```
6
```

Sample Input file: input_2.txt

```
3
2
```

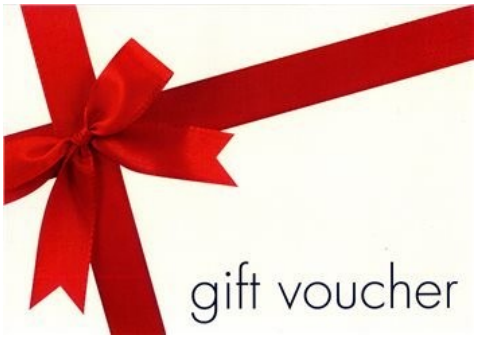
Output to be printed:

```
ERROR
```

P2. Let's Use Our Voucher!

Problem Description.

Cappuccino, a friend of Latte and Mocca, is happy of seeing them finally together, and has bought them a Saint Valentine's voucher to spend at Geek Thomas, their favourite store.



Geek Thomas has a range of product and prices <P1, P2, …, Pn>. The voucher has value T. Latte and Mocca are sharing the voucher by buying 1 product each (Latte buys Pi and Mocca buys Pj, with Pi != Pj). To make the most out of the voucher, they want to buy the two products Pi and Pj whose sum gets closer to T.

Your task is to...
Help Latte and Mocca to decide which two products Pi and Pj to buy, and then print by console the absolute difference between the sum of the chosen Pi + Pj and T.

More precisely, you will read T and <P1, P2, …, Pn> from a file f. For each pair Pi and Pj their distance to T can be computed as the absolute value of |T – (Pi + Pj)| = d.
You need to print the minimum distance d found.

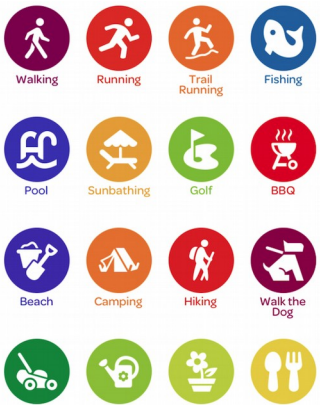
Sample input_1.txt		Output	Explanation: Distances d	
8	*(T)*	2	P1+ P2 => 1 + 4 = 5	| T - 5 | => | 8 - 5 | = 3
1	*(P1)*		P1 + P3 => 1 + 9 = 10	| T - 10 | => | 8 - 10 | = 2
4	*(P2)*		P2 + P3 => 9 + 4 = 13	| T - 13 | => | 8 - 13 | = 5
9	*(P3)*			

Sample input_2.txt		Output	Explanation: Distances d	
14	*(T)*	1	P1 + P2 => 2 + 10 = 12	| T - 12 | => | 14 - 12 | = 2
2	*(P1)*		P1 + P3 => 2 + 5 = 7	| T - 7 | => | 14 - 7 | = 7
10	*(P2)*		P1 + P4 => 2 + 7 = 9	| T - 9 | => | 14 - 9 | = 5
5	*(P3)*		P2 + P3 => 10 + 5 = 15	| T - 15 | => | 14 - 15 | = 1
7	*(P4)*		P2 + P4 => 10 + 7 = 17	| T - 17 | => | 14 - 17 | = 3

P3. Are We a Formal Ewok Couple?

Problem Description.

Time has passed and the dates of Latte and Mocca are going very well. To such extent that they are start wondering whether they can call themselves a *formal Ewok couple*. This is indeed much more serious than it seems at first sight. Ewoks take very seriously the term *formal couple:* for two of these fascinating creatures to claim themselves as such, a number of activities have to be completed by them first (including going fishing, camping or rescuing Han Solo & Pricess Leia from some stormtroopers).



In this problem we simulate the activities performed by a couple of Ewoks by making an analogy with the letters of the alphabet contained in a sentence.

Thus, the alphabet contains 26 letters, which represent the potential 26 different activities a couple of Ewoks have to complete before calling themselves a *formal couple*. In this context, letters ‘a’ and ‘A’ stand for the activity going fishing, letters ‘b’ and ‘B’ stand for the activity going camping, and so on).

The activities completed by an Ewok couple is provided via a sentence with N characters, where the each character represents the activity the Ewoks did during their i-est date.

Your task is to...

Help Latte and Mocca to decide whether they are already a *formal Ewok couple*. To do so, you need to read a sentence s from file f and print by the screen how many letters l of the alphabet it contains. If the answer is 26, then the Ewoks are formal enough, otherwise they are not.

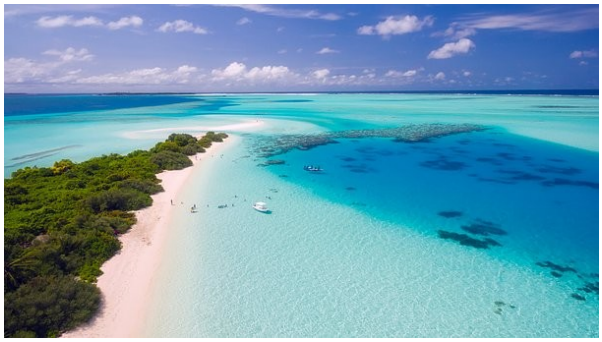
Sample input 1.txt	Output
Hello, good morning, how are you?	14

Sample input 2.txt	Output
The quick brown fox Jumps over the lazy dog!	26

P4. Let's Explore as Many Planets as We Can?

Problem Description.

Latte and Mocca have finally reached the status of *formal Ewok couple*. This includes, among other benefits, a visit to *Americano*, an oracle foreseeing their future travelling possibilities.



- Let D be a list representing the concrete days required to visit each of the N planets of the galaxy D = <D1, D2, ..., Dn>.
  For example, D = <2, 6, 4> mean that the planets 1, 2 and 3 require 2, 6 and 4 days to visit them (resp.)
- Let H = <H1, H2, ..., Hn> be a list representing the concrete days of holidays foreseen in the next N years.
  For example, H = <5, 1, 3> mean that the years 1, 2 and 3 have 5, 1 and 3 days of holidays (resp.)
- A planet Dj is **properly explored** if visited in a year Hi with enough holidays (Hi >= Dj).
  For example, if planet D3 is visited in year H1 then it is properly explored, as 5 >= 4.
  However, if the planet D3 is visited in year H2 then it is not properly explored, as 1 < 4.

Latte and Mocca are decided: they will visit all N planets, one different planet each year. To decide the year they will visit each planet, they compute a list A = <A1, A2, ..., An>.

- *For example, a computed list A = <2, 3, 1> mean D1, D2 and D3 are visited on years H2, H3 and H1, resp.*
  - *D1 is not properly visited, as H2 < D1 => 1 < 2.*
  - *D2 is not properly visited, as H3 < D2 => 3 < 6.*
  - *D3 is properly visited, as H1 >= D3 => 5 >= 4.*

  So, all in all, the number of planets properly visited is 1 (just D3).
- *For example, a computed list A = <3, 2, 1> mean D1, D2 and D3 are visited on years H3, H2 and H1, resp.*
  - *D1 is properly visited, as H3 >= D1 => 3 >= 2.*
  - *D2 is not properly visited, as H2 < D2 => 1 < 6.*
  - *D3 is properly visited, as H1 >= D3 => 5 >= 4.*

  So, all in all, the number of planets properly visited is 2 (D1 and D2).

Your task is to...

Help Latte and Mocca to create the optimal assignment A maximising the number of planets properly explored. Print by console such assignment, together with the amount of planets properly explored.

Sample input_1.txt		Output
3	(N)	2
2	(D0)	[2, 1, 0]
6	(D1)	
4	(D2)	
5	(H0)	
1	(H1)	
3	(H2)	

Explanation: The optimal assignment A = <2, 1, 0>
- Properly explores D0 on year H2
- Does not properly explore D1 on year H1
- Properly explores D2 on year H0

So, all in all, 2 planets are properly explored.

Sample input_2.txt		Output
5	(N)	3
9	(D0)	[0, 2, 4, 1, 3]
5	(D1)	
3	(D2)	
7	(D3)	
10	(D4)	
1	(H0)	
8	(H1)	
6	(H2)	
2	(H3)	
4	(H4)	

Explanation: The optimal assignment A = <0, 2, 4, 1, 3>
- Does not properly explores D0 on year H0
- Properly explores D1 on year H2
- Properly explores D2 on year H4
- Properly explores D3 on year H1
- Does not properly explore D4 on year H3

So, all in all, 3 planets are properly explored.

P5. Are we a Formal and Stable Ewok Couple?

Problem Description.

After a few years travelling accross the planets of the galaxy as a *formal Ewok couple*, its time for Latte and Mocca to make *the big step*. Espresso, the ancient and most knowledgeable of Ewoks tells our young couple about the last test for them to pass: *the Stable Ewok Couple Problem*.
The problem is defined as follows:

- Let G1 = <F1, F2, ..., FN> and G2 = <M1, M2, ..., MN> be two groups of N Ewoks each.
  For example G1 = *<1 (Latte), 2 (Macchiato)>* and *G2 = <1 (Mocca), 1 (Affogato)> means there are two groups of Ewoks of N = 2 members each. For a better understanding of the rest of the example, rather than calling these members by their actual names (1) and (2), we will call them by their nicknames Latte, Macchiato, Mocca and Affogato.*

- Let P_G1 =

Member	Pref1	Pref2	....	PrefN
< F1	M11	M12	...	M1N >
< F2	M21	M22	...	M2N >
...	...	....	...	....
< FN	MN1	MN2	...	MNN >

and

Let P_G2 =

Member	Pref1	Pref2	....	PrefN
< M1	F11	F12	...	F1N >
< M2	F21	F22	...	F2N >
...	...	....	...	...
< MN	FN1	FN2	...	FNN >

be two 2D-matrix stating the preferences of each member of G1 over the members of G2 (and viceversa). Please note each member ranks the N members of the other group, with Pref1 being the most preffered member, Pref2 being the second preffered member, and so on.

For example P_G1 = < *<1, 2>* => *Latte prefers 1st Mocca and 2nd Affogato*
<1, 2> > => *Macchiato prefers 1st Mocca and 2nd Affogato*

For example P_G2 = < *<1, 2>* => *Mocca prefers 1st Latte and 2nd Macciato*
<2, 1> > => *Affogato prefers 1st Macciato and 2nd Latte*

- Let A = <A1, A2, …, An> be a list matching all members of G1 and G2 as couples. Each A[i] represents the member M' of G2 assigned to Fi.

  For example, the assignment A' = <2, 1> means that
  - *M2 (Affogato) has been matched to F1 (Latte);*
    M1 (Mocca) has been matched to F2 (Machiato);

  For example, the assignment A = <1, 2> means that
  - *M1 (Mocca) has been matched to F1 (Latte);*
  - *M2 (Affogato) has been matched to F2 (Machiato);*

- Let A be a **non-stable assignment** if at least 1 members of G1 and G2 are not happy with their current couples and prefer to break-up with them to match themselves.
  For example, the assignment A' = <2, 1> is non-stable.
  Let's see the situation of our four members:
  - *Latte prefers 1st Mocca and 2nd Affogato. Latte is matched to Affogato*
  - *Macchiato prefers 1st Mocca and 2nd Affogato. Macchiato is matched to Mocca*
  - *Mocca prefers 1st Latte and 2nd Macciato. Mocca is matched to Macchiato*
  - *Affogato prefers 1st Macciato and 2nd Latte. Affogato is matched to Latte.*

To make the assignment non-stable we need at least 1 members of G1 and G2 are not happy with their current couples and prefer to break-up with them to match themselves. This is precisely the case of Latte of Mocca:

- Latte is assigned to Affogato, but it prefers Mocca instead.
  Mocca is assigned to Macchiato, but it prefers Latte instead.
  So, both Latte and Mocca will ask for a new assignment matching them as a couple.

Formally, to check for non-stable couples Fi-Mj one has to traverse the rank of Fi:

1. Is there any other Ekow M' than Fi prefers to its current couple Mj?
2. If so, what about M'? Would this Ewok prefer Fi to its current couple too?

If we found 1 and 2 hold, then these two Ewoks will make our assignment non-stable.

- Let A be a **stable assignment** if it is break-up free.
  For example, the assignment A = <1, 2> is stable.
  Let's see the situation of our four members:
  - *Latte prefers 1st Mocca and 2nd Affogato. Latte is matched to Mocca*
  - *Macchiato prefers 1st Mocca and 2nd Affogato. Macchiato is matched to Afffogato*
  - *Mocca prefers 1st Latte and 2nd Macciato. Mocca is matched to Latte*
  - *Affogato prefers 1st Macciato and 2nd Latte. Affogato is matched to Macchiato.*

While this assignment doesn't make everybody happy, it is stable.
Macchiato will want to break-up with Affogato and go with Mocca, but this won't happen, as Mocha doesn't prefer Macchiato to its current couple.

Your task is to...
Compute A to be a stable assignment.

Sample input_1.txt

```
2          (N)
1    2     (P_G1  F1 )
1    2     (P_G1  F2)
1    2     (P_G2  M1)
2    1     (P_G2  M2)
```

Output

(1, 1), (2, 2)

Explanation: The stable assignment A = <1, 2> matches:
- The Ewok F1 with M1.
- The Ewok F2 with M2.

Sample input_2.txt

```
4                        (N)
2    4    1    3        (P_G1  F1)
3    1    4    2        (P_G1  F2)
2    3    1    4        (P_G1  F3)
4    1    3    2        (P_G1  F4)
2    4    1    3        (P_G2  M1)
4    3    1    2        (P_G2  M2)
1    4    3    2        (P_G2  M3)
2    1    4    3        (P_G2  M4)
```

Output

(1, 4), (2, 1), (3, 2), (4, 3)
(1, 4), (2, 3), (3, 2), (4, 1)

Explanation: The stable assignment A = <4, 1, 2, 3> matches:
- The Ewok F1 with M4.
- The Ewok F2 with M1.
- The Ewok F3 with M2.
- The Ewok F4 with M3.

Another stable assignment would be A’ = <4, 3, 2, 1>, which matches:
- The Ewok F1 with M4.
- The Ewok F2 with M3.
- The Ewok F3 with M2.
- The Ewok F4 with M1.
