Q1_Explain the changes that would have to be made to the program of Code Fragment 3.8 so that it could perform the Caesar cipher for messages that are written in an alphabet-based language other than English, such as Greek, Russian, or Hebrew.

التعديلات المطلوبة:

1تغيير حجم الأبجدية

- في النسخة الأصلية، الأبجدية الإنجليزية تتكون من **26حرفًا** (`A-Z`أو `a-z`).
- في اللغات الأخرى، قد يختلف عدد الأحرف:
  - **اليونانية** 24 :حرفًا (A-Ω ، α-ω)
  - **الروسية** 33 :حرفًا (A-Я ، a-я)
  - **العبرية** 22 :حرفًا (א-ת)
- يجب تعديل الكود ليستخدم الحجم الصحيح لكل أبجدية.

2تحديد نطاق الأحرفUnicode

- الأحرف في الإنجليزية تقع ضمن **نطاق ASCII** (`A-Z` = 65-90 و`a-z` = 97-122).
- الأبجديات الأخرى تستخدم **Unicode**، لذا يجب ضبط تعويض التشفير بناءً على **نطاق الأحرف الخاصة باللغة**.

مثال على تحديد النطاقات:

- **اليونانية** (Greek):

```java
char base = Character.isUpperCase(ch) ? 'A' : 'α';
int alphabetSize = 24;
```

- **الروسية** (Russian):

```java
char base = Character.isUpperCase(ch) ? 'A' : 'a';
int alphabetSize = 33;
```

- **العبرية** (Hebrew):
  - لا تفرق بين الحروف الكبيرة والصغيرة.

```java
char base = 'א';
int alphabetSize = 22;
```

3تعديل معادلة التشفير والفك

المعادلة الأصلية لشيفرة قيصر هي:

$$C=(P+k) \mod N \quad C = (P + k) \mod N \quad C=(P+k) \mod N$$

حيث:

- C هو الحرف المشفر.
- P هو الحرف الأصلي.
- k هو مقدار الإزاحة (المفتاح).
- N هو عدد الحروف في الأبجدية.

لضبط الكود ليتناسب مع الأبجديات المختلفة، يجب تعديل العملية كما يلي:

```
java
نسختحرير
char encryptedChar = (char) ((ch - base + shift) % alphabetSize +
                             base);
```

وإذا كان `ch` خارج النطاق الأبجدي (مثل الرموز أو الأرقام)، يتم **تخطي التشفير** للحفاظ على النص الأصلي.

---

4 معالجة الأحرف غير الأبجدية

- بعض اللغات (مثل العبرية) لا تحتوي على أحرف صغيرة/كبيرة، لذا لا داعي لمعالجة الفرق بينها.
- يجب التأكد من **عدم تشفير الأرقام والرموز الخاصة**.

Q2_What is the difference between a shallow equality test and a deep equality test between two Java arrays, A and B, if they are one-dimensional arrays of type int? What if the arrays are two-dimensional arrays of type int?

1 في حالة المصفوفات أحادية البعد (One-Dimensional int Arrays)

📌 المساواة السطحية (Shallow Equality):

- يتم التحقق مما إذا كانت **المصفوفتان تشير إلى نفس الكائن في الذاكرة**، وليس ما إذا كانت القيم بداخلهما متطابقة.
- يتم ذلك باستخدام **عامل المساواة (==)**.
- إذا كانت `A == B` فهذا يعني أن كلاهما يشير إلى **نفس المصفوفة في الذاكرة**.

📌 المساواة العميقة (Deep Equality):

- يتم التحقق مما إذا كانت **جميع القيم داخل المصفوفتين متطابقة** وليس فقط مرجعيهما في الذاكرة.
- يتم ذلك باستخدام `Arrays.equals(A, B)`.

2 في حالة المصفوفات ثنائية البعد (Two-Dimensional int Arrays)

📌 المساواة السطحية (Shallow Equality):

- إذا استخدمنا `A == B` في الذاكرة A وB، فسيتم التحقق مما إذا كانت **نفس كائن المصفوفة الثنائية** تشير إلى.
- إذا كانت كل مصفوفة كائناً منفصلاً `false` متطابقة، فستُرجع A وB ولكن حتى لو كانت القيم داخل.

📌 المساواة العميقة (Deep Equality):

- لا يمكن استخدام `Arrays.equals(A, B)` لأنه يتحقق فقط من المساواة بين **المراجع** وليس القيم.
- بدلاً من ذلك، يجب استخدام `Arrays.deepEquals(A, B)`، الذي يتحقق من القيم داخل كل صف.

Q3_ Give three different examples of a single Java statement that assigns variable, backup, to a new array with copies of all int entries of an existing array, original.

الطريقة ١: باستخدام `Arrays.copyOf`

- هذه الطريقة تنشئ مصفوفة جديدة بنفس طول `original` وتنسخ جميع القيم إليها.

✅**الطريقة ٢: باستخدام `clone`**

- هذه الطريقة تنشئ نسخة عميقة جديدة من `original` إذا كانت مصفوفة أحادية البعد.

✅**الطريقة ٣: باستخدام `System.arraycopy`**

- تستخدم هذه الطريقة لنسخ أجزاء محددة من المصفوفة بسرعة وكفاءة.
- **◆مقارنة الطرق الثلاثة**

الطريقة	هل تنسخ القيم؟	متى تستخدم؟
`Arrays.copyOf`	✅نعم	الأسهل للاستخدام
`clone`	✅نعم	الأفضل للمصفوفات أحادية البعد
`System.arraycopy`	✅نعم	الأفضل للأداء عند التعامل مع كميات كبيرة من البيانات

لاب ٣