Taak 2.2 – Annihilatie (annihilation) (100 ptn)

In de deeltjesfysica bestaat er voor elk deeltje een tegengesteld antideeltje. Zo is het antideeltje van een elektron bijvoorbeeld een positron, en omgekeerd. Wanneer een deeltje en zijn antideeltje met twee bij elkaar komen, verdwijnen beide, en komt er een hoeveelheid energie vrij[1] en spreken we van *annihilatie*. Beschikkende over een hele verzameling verschillende deeltjes en antideeltjes, wil de *belgian energy Creation and Production* deze graag gebruiken om energie mee op te wekken (wat geen schitterend idee is, omdat je hiervoor véél deeltjes nodig hebt). Om winstgevend te blijven, kunnen ze maximaal K van deze deeltjes ongebruikt laten.

Het productieproces gaat als volgt. Er worden N deeltjes en antideeltjes in afgescheiden vakjes op een rij naast elkaar gelegd. De scheidingsmuren van de vakjes worden vervolgens verwijderd en de resulterende energie, ontstaan uit de annihilatie van naast elkaar liggende deeltjes, wordt opgevangen. Wanneer er deeltjes annihileren geeft dat de kans aan de omliggende deeltjes om verder ook te annihileren. Dit proces moet in een streng beveiligde en vacuüm omgeving gebeuren.

Halverwege de uitvoering van deze procedure loopt het mis: niet alle aanliggende deeltjes blijken noodzakelijk met elkaar te kunnen annihileren. Manueel ingrijpen is omwille van de veiligheidsvoorschriften natuurlijk onmogelijk, maar gelukkig beschikt de firma over een ionen-kanon. Daarmee kunnen ze per schot een enkel vakje uit de rij weghalen, zodat de omliggende vakjes naast elkaar komen te liggen. Ze willen natuurlijk ook geen verlies lijden, dus willen ze het aantal schoten S zou klein mogelijk maken. Als de minimale S groter dan K zou blijken, moet de hele procedure geannuleerd en heropgestart worden.

Taak

Schrijf een programma dat, gegeven de volgorde van de N beschikbare deeltjes en antideeltjes en de waarde van K, de minimale S berekent indien de procedure kan verdergaan na eventueel gebruik van het kanon, en anders -1 teruggeeft, als de procedure stopgezet moet worden.

Limieten en beperkingen

- $1 \leq N \leq N_{MAX}$, het aantal deeltjes en antideeltjes;
- $0 \leq K \leq K_{MAX}$, het aantal (anti)deeltjes dat maximaal verloren mag gaan;

	N_{MAX}	K_{MAX}
Subtaak A (7 ptn)	400	1
Subtaak B (35 ptn)	1 000	1
Subtaak C (42 ptn)	1 000 000	1
Subtaak D (16 ptn)	400	400

Maximale uitvoeringsduur: **1 seconde**. Geheugenlimiet: **64 MB**.

Opgepast: *Zoals je kan zien zijn de subtaken behoorlijk uiteenlopend. Je score wordt echter bepaald door* ***je submissie met maximale score****, en niet door de som van de punten van de subtaken die je doorheen de wedstrijd opgelost hebt. Daarom kan het nodig zijn de bovenstaande beperkingen te testen op de input en naargelang daarvan de code die je programma uitvoert aan te passen.*

Input

Jouw programma krijgt input in het volgende formaat:

- een getal K, het maximaal aantal deeltjes dat verloren mag gaan
- Een enkele lijn met een string van N deeltjes

[1] volgens de bekende formule van Einstein: $E = mc^2$

Een deeltje is een enkel karakter van het alfabet. Een deeltje a is een antideeltje van b als en slechts als a en b dezelfde letter zijn met a een hoofdletter en b een kleine letter of omgekeerd.

`X` en `x` zijn dus antideeltjes van elkaar.

Output

Jouw programma moet naar de output de minimale waarde van S schrijven, of -1 als de procedure gestopt moet worden, gevolgd door een nieuwe lijn.

Voorbeeld 1

Gegeven de volgende input:

```
1
aABb
```

moet jouw programma het volgende teruggeven:

```
0
```

Hier annihileren tegelijkertijd de 'a' met de 'A' en de 'B' met de 'b'. Het ionen-kanon hoeft dus niet gebruikt te worden.

Voorbeeld 2

Gegeven de volgende input:

```
1
BaAab
```

moet jouw programma het volgende teruggeven:

```
1
```

Als we een van de 'a's weghalen, kan de 'A' met de overblijvende 'a' annihileren, waarna de 'B' hetzelfde kan doen met de 'b' (omdat deze nu naast elkaar komen te liggen). Het minimaal aantal schoten is dus gelijk aan 1.

Voorbeeld 3

Gegeven de volgende input:

```
1
beCP
```

moet jouw programma het volgende teruggeven:

```
-1
```

Geen enkel paar deeltjes kan met elkaar annihileren, dus zouden we hier 4 schoten moeten lossen. Omdat we slechts het verlies van een enkel deeltje toestaan, moeten we de procedure dus stopzetten.