Password security is a tricky issue. Users usually prefer short and simple passwords that are easy to remember, but such passwords are not secure enough. So, there should be some password policy requirements in security-enhanced environments. Below is an example of a password policy requirement for a user account in a domain:

1. The password should be at least six characters in length
2. It should contain at least one English uppercase letter (A through Z)
3. It should contain at least one English lowercase letter (a through z)
4. It should contain at least one base 10digit (0 through 9)

Given a string of alphanumeric (lowercase, uppercase, or digit) characters, your task is to find the shortest contiguous substring which satisfies the password policy requirement given above.

If there are several possible answers, return the one that appears in the given string first. If there is no answer, return empty string ("") instead.

**Example:**

PasswordPolicy("code2016Fights") = "e2016F"

* **[input] string Password**
  + Password.length ≤ 100
* **[output] string**

<https://codefights.com/challenge/BMaKJx4KeWKisHjzp>

static string PasswordPolicy(string Password)

{

string ans = "";

for (int i = 0; i < Password.Length; i++)

{

//for (int j = Password.Length - 1; j > i + 1; j--)

for(int j =i+2; j < Password.Length; j++)

{

string subs = Password.Substring(i, j - i+1);

//Console.WriteLine(subs);

bool mayus = false, minus = false, dig = false;

for (int k = 0; k < subs.Length; k++)

{

if ('A' <= subs[k] && subs[k] <= 'Z')

{

mayus = true;

}

if ('a' <= subs[k] && subs[k] <= 'z')

{

minus = true;

}

if ('0' <= subs[k] && subs[k] <= '9')

{

dig = true;

}

}

if (mayus && minus && dig)

{

if (subs.Length >= 6 && ans.Length == 0)

{

ans = subs;

}

else if (subs.Length >= 6 && ans.Length > 0)

{

if (subs.Length < ans.Length)

{

ans = subs;

}

}

}

}

}

return ans;

}