```cpp
#include <bits/stdc++.h>
using namespace std;
#define ll long long
```

Prefix function KMP for string:

```cpp
vector<int> pi(100100);
string s;

void prefix_function() {
    int n = (int)s.size();
    for (int i = 1; i < n; i++) {
        int j = pi[i-1];
        while (j > 0 && s[i] != s[j])
            j = pi[j-1];
        if (s[i] == s[j])
            j++;
        pi[i] = j;
    }
}
```

Prefix function KMP for numbers:

```cpp
vector<int> pi(100100) , v(100100);

void prefix_function() {
    int n = (int)v.size();
    for (int i = 1; i < n; i++) {
        int j = pi[i-1];
        while (j > 0 && v[i] != v[j])
            j = pi[j-1];
        if (v[i] == v[j])
            j++;
        pi[i] = j;
    }
}
```

KMP Applications:

```cpp
// 1. find and display the positions of all occurrences of the string s in the string t by s+#+t;

// 2. Counting the number of occurrences of each prefix:
// in the same string use the function
// in different strings use the same function but start the iteration from s.size()+1
vector<int> pi(100100) , ans(100200);

void countPrefix(){
    int n = (int)pi.size();
    for (int i = 0; i < n; i++)
        ans[pi[i]]++;
    for (int i = n-1; i > 0; i--) // this is reverse for the first one
        ans[pi[i-1]] += ans[i];
    for (int i = 0; i <= n; i++) // this is same as the first one
        ans[i]++;
}

// 3. Compressing a string()
// compressing is string t of smallest length such that s can be represented
// as a concatenation of one or more copies of t

// we calc the value k = n-pi[n-1], if k divides n, then k will be the answer,
// otherwise there is no effective compression and the answer is n
```