LU_dAREdevils
Leading University

String:

Double Hashing:

```
const int N = 1e6 + 5;
const int Base1 = 137, Base2 = 277;
const int mod1 = 127657753, mod2 = 987654319;

bool isCalPow = 0;
pair<ll, ll> po[N];
void generatePower() // Storing the power of the Base.
{
  po[0].first = 1, po[0].second = 1;
  for (int i = 1; i < N; i++) {
    po[i].first = (po[i - 1].first * Base1) % mod1;
    po[i].second = (po[i - 1].second * Base2) % mod2;
  }
}
struct Hashing {
  vector<pair<ll, ll>> prefix, suffix;
  int n;
  void generatePrefixHash(string &s) {
    prefix[0].first = s[0], prefix[0].second = s[0];
    for (int i = 1; i < s.size(); i++) {
      prefix[i].first = ((prefix[i - 1].first * Base1) + s[i]) % mod1;
      prefix[i].second = ((prefix[i - 1].second * Base2) + s[i]) % mod2;
    }
  }
  void generateSuffixHash(string &s) {
    suffix[n - 1].first = s[n - 1], suffix[n - 1].second = s[n - 1];
    for (int i = n - 2; i >= 0; i--) {
      suffix[i].first = ((suffix[i + 1].first * Base1) + s[i]) % mod1;
      suffix[i].second = ((suffix[i + 1].second * Base2) + s[i]) % mod2;
    }
  }
  pair<ll, ll> generateHash(string &s) // return hash value of a string
  {
    pair<ll, ll> H = {0, 0};
    for (auto &c : s)  {
      H.first = ((H.first * Base1) + c) % mod1;
      H.second = ((H.second * Base2) + c) % mod2;
    }
    return H;
  }
  pair<ll, ll> getPrefixRangeHash(int l, int r) // return hash value of a range
  {
    if (l == 0) return prefix[r];
    pair<ll, ll> Hs;
    Hs.first = (prefix[r].first - (prefix[l - 1].first * po[r - l + 1].first % mod1) + mod1) % mod1;
    Hs.second = (prefix[r].second - (prefix[l - 1].second * po[r - l + 1].second % mod2) + mod2) % mod2;
    return Hs;
  }
  pair<ll, ll> getSuffixRangeHash(int l, int r) // return hash value of a range
  {
    if (r == n - 1) return suffix[l];
    pair<ll, ll> Hs;
    Hs.first = (suffix[l].first - (suffix[r + 1].first * po[r - l + 1].first % mod1) + mod1) % mod1;
```