```
namespace __hash__ {
  const int mds_cnt = 10;
  const int use_cnt =  2;
  const int base = 727;
  const int Maxx = 1e5 + 6.66;
  int mods[mds_cnt] = {1000000007, 1000000009, 1000000021, 1000000033,
   1000000087,
    1000000093, 1000000097, 1000000103, 1000000123, 1000000181};
  int in_use[use_cnt];
  bool initiated = false;
  bool initiated2 = false;

  void init() {
    if(initiated) return ;
    initiated = true;
    srand(time(0));
    for(int i = 0; i < use_cnt; i++) {
      bool ok = false;
      while(!ok) {
        in_use[i] = rand() % mds_cnt;
        ok = true;
        for(int j = 0; j < i; j++) if(in_use[i] == in_use[j]) ok = false;
      }
    }
    for(int i = 0; i < use_cnt; i++) in_use[i] = mods[in_use[i]];
  }

  struct Hash {
    int num[use_cnt];

    Hash() {
      init();
      memset(num, 0, sizeof num);
    }

    Hash(long long x) {
      init();
      for(int i = 0; i < use_cnt; i++) num[i] = x % in_use[i];
    }

    Hash(long long x, long long nx) {
      init();
      for(int i = 0; i < use_cnt; i++) {
        num[i] = x % in_use[i];
        num[i] = (1ll * num[i] * base % in_use[i] + nx) % in_use[i];
      }
    }

    Hash(string str) {
      init();
```

```
  memset(num, 0, sizeof num);
  for(auto ch : str)
    *this *= base, *this += (int) ch;
}

Hash& operator+=(Hash x) {
  for(int i = 0; i < use_cnt; i++) {
    num[i] = (num[i] + x.num[i]);
    if(num[i] >= in_use[i]) num[i] -= in_use[i];
  }
  return *this;
}

Hash& operator-=(Hash x) {
  for(int i = 0; i < use_cnt; i++)
  {
    num[i] -= x.num[i];
    if(num[i] < 0) num[i] += in_use[i];
  }
  return *this;
}

Hash& operator*=(Hash x) {
  for(int i = 0; i < use_cnt; i++)
    num[i] = 1ll * num[i] * x.num[i] % in_use[i];
  return *this;
}

Hash operator+(Hash y) {
  Hash w = *this;
  return w += y;
}

Hash operator-(Hash y) {
  Hash w = *this;
  return w -= y;
}

Hash operator*(Hash y) {
  Hash w = *this;
  return w *= y;
}

bool operator==(Hash y) {
  for(int i = 0; i < use_cnt; i++)
    if(y.num[i] != num[i]) return false;
  return true;
}

bool operator<(Hash y) const {
  for(int i = 0; i < use_cnt; i++) {
```

```
      if(y.num[i] == num[i]) continue;
      return num[i] < y.num[i];
    }
    return false;
  }

  Hash pow(int exp) {
    Hash tmp = *this;
    Hash ret = 1;
    for(; exp; exp >>= 1, tmp *= tmp)
      if(exp & 1)
      {
        ret *= tmp;
      }
    return ret;
  }
} pow_base[Maxx];

void init2() {
  if(initiated2) return ;
  init();
  initiated2 = true;
  pow_base[0] = 1;
  for(int i = 1; i < Maxx; i++)
  {
    pow_base[i] = pow_base[i - 1] * base;
  }
}

struct String {
  vector<Hash> ss;

  String() { }

  String(string str) {
    init2();
    ss = vector<Hash>(str.size() + 1);
    for(int i = 0; i < str.size(); i++) {
      if(i > 0) {
        ss[i + 1] = ss[i];
        ss[i + 1] *= base;
      }
      ss[i + 1] += (int) str[i];
    }
  }

  Hash get_substring(int l, int r) { // [l, r)
    return ss[r] - ss[l] * pow_base[r - l];
  }
};
};
```