**Miller-Rabin primality test**

**using u64 = uint64\_t;**

**using u128 = \_\_uint128\_t;**

**u64 binpower(u64 base, u64 e, u64 mod) {**

**u64 result = 1;**

**base %= mod;**

**while (e) {**

**if (e & 1)**

**result = (u128)result \* base % mod;**

**base = (u128)base \* base % mod;**

**e >>= 1;**

**}**

**return result;**

**}**

**bool check\_composite(u64 n, u64 a, u64 d, int s) {**

**u64 x = binpower(a, d, n);**

**if (x == 1 || x == n - 1)**

**return false;**

**for (int r = 1; r < s; r++) {**

**x = (u128)x \* x % n;**

**if (x == n - 1)**

**return false;**

**}**

**return true;**

**};**

**bool MillerRabin(u64 n, int iter=5) { // returns true if n is probably prime, else returns false.**

**if (n < 4)**

**return n == 2 || n == 3;**

**int s = 0;**

**u64 d = n - 1;**

**while ((d & 1) == 0) {**

**d >>= 1;**

**s++;**

**}**

**for (int i = 0; i < iter; i++) {**

**int a = 2 + rand() % (n - 3);**

**if (check\_composite(n, a, d, s))**

**return false;**

**}**

**return true;**

**}**

### Deterministic version

**bool MillerRabin(u64 n) { // returns true if n is prime, else returns false.**

**if (n < 2)**

**return false;**

**int r = 0;**

**u64 d = n - 1;**

**while ((d & 1) == 0) {**

**d >>= 1;**

**r++;**

**}**

**for (int a : {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37}) {**

**if (n == a)**

**return true;**

**if (check\_composite(n, a, d, r))**

**return false;**

**}**

**return true;**

**}**