

```

63 Calculate the value of  $(x^n) \% m$  :
64 int modPow(int x, int n, int m) {
65     if (n == 0) return 1 % m;
66     ll u = modPow(x, n/2, m);
67     u = (u * u) % m;
68     if (n%2 == 1) u = (u * x) % m;
69     return u;
70 }
71
72 Operation functions with mod:
73 int mul(int x, int y, int m)
74 {
75     return (ll) x*y % m;
76 }
77 int sum(int x, int y, int m)
78 {
79     return (x+y) % m;
80 }
81 int sub(int x, int y, int m)
82 {
83     return sum((x-y)%m, m, m);
84 }
85 int po(int x, int y, int m)
86 {
87     if(!y) return 1;
88     if(y&1) return mul(x, po(x,y-1,m), m);
89     int z = po(x, y/2, m);
90     return mul(z, z, m);
91 }
92 int inv(int x, int m)
93 {
94     return po(x, m-2, m);
95 }
96 // note:  $3/5 \% MOD \implies 3*inv(5, MOD)$ 
97
98 Operation function with mod:
99 //this function faster then the other
100 void add_self(int& x, int y)
101 {
102     if((x += y) ≥ MOD) x -= MOD;
103 }
104 int add(int x, int y)
105 {
106     return add_self(x, y), x;
107 }
108 void sub_self(int& x, int y)
109 {
110     if((x -= y) < 0) x += MOD;
111 }
112 int sub(int x, int y)
113 {
114     return sub_self(x, y), x;
115 }
116 int mul(int x, int y)
117 {
118     return (long long) x * y % MOD;
119 }

```