```
class GNUM
{
private:
        static const int BASE = 10;
        static const int BASE_LEN = 1;
        static const int LEN = 20100;
        int a[LEN];
        int len;

public:
        GNUM()
        {
                len = 0;
                memset(a, 0, sizeof(a));
        }
        ~GNUM()
        {
                len = 0;
                memset(a, 0, sizeof(a));
        }
        template <typename T>
        GNUM(T x)
        {
                len = 0;
                memset(a, 0, sizeof(a));
                if (x == 0)
                {
                        a[++len] = x;
                        return;
                }
                while (x)
                {
                        a[++len] = x % BASE;
                        x /= BASE;
                }
        }
        template <typename Y>
        GNUM(Y *s, int lenth)
        {
                len = lenth;
                memcpy(a, s, sizeof(a));
        }
        GNUM(char *x)
        {
                len = 1;
                register int lenth = strlen(x) - 1;
                memset(a, 0, sizeof(a));
                for (register int i = lenth, j = 1; i >= 0; --i)
                {
                        if (j == BASE)
                                j = 1, ++len;
                        a[len] += (x[i] - '0') * j;
                        j *= 10;
                }
        }
        friend istream &operator>>(istream &input, GNUM &in)
        {
                char s[LEN];
                input >> s;
                in = GNUM(s);
                return input;
        }
```

```
friend ostream &operator<<(ostream &output, GNUM out)
{
        output << out.a[out.len];
        for (register int i = out.len - 1; i; --i)
        {
                output << right << setw(BASE_LEN) <<

                        setfill('0') << out.a[i];
        }
        return output;
}
inline void print()
{
        printf("%d", a[len]);
        for (register int i = len - 1; i; --i)
                printf("%04d", a[i]);
}
inline GNUM del_zero()
{
        while (!a[len] && len > 1)
                --len;
}
inline void add(int k)
{
        if (!k && len == 1 && !a[1])
                return;
        for (register int i = len; i; --i)
                a[i + 1] = a[i];
        a[1] = k;
        ++len;
}
inline friend bool operator<(const GNUM &x, const GNUM &y)
{
        if (x.len == y.len)
        {
                register int i;
                for (i = x.len; x.a[i] == y.a[i] && i > 1; --i)
                        ;
                if (i >= 1)
                        return x.a[i] < y.a[i];
                else
                        return false;
        }
        else
                return x.len < y.len;
}
inline friend bool operator>(const GNUM &x, const GNUM &y)
{
        if (x.len == y.len)
        {
                register int i;
                for (i = x.len; x.a[i] == y.a[i] && i > 1; --i)
                        ;
                if (i >= 1)
                        return x.a[i] > y.a[i];
                else
                        return false;
        }
        else
                return x.len > y.len;
        return false;
}
```

```
inline friend bool operator==(GNUM x, GNUM y)
{
        if (x.len == y.len)
        {
                register int i;
                for (i = x.len; x.a[i] == y.a[i] && i >= 1; --i)
                        ;
                if (i >= 1)
                        return false;
                return true;
        }
        else
                return false;
        return false;
}
inline friend bool operator<=(const GNUM &x, const GNUM &y)
{
        return !(x > y);
}
inline friend bool operator>=(const GNUM &x, const GNUM &y)
{
        return !(x < y);
}
template <typename AF>
inline friend GNUM operator+(const GNUM &x, const AF &y)
{
        return x + GNUM(y);
}
inline friend GNUM operator+(const GNUM &x, const GNUM &y)
{
        GNUM res;
        res.len = max(x.len, y.len);
        int k = 0;
        for (register int i = 1; i <= res.len; ++i)
        {
                res.a[i] = x.a[i] + y.a[i] + k;
                k = res.a[i] / BASE;
                res.a[i] %= BASE;
        }
        if (k)
                res.a[++res.len] = k;
        res.del_zero();
        return res;
}
template <typename JF>
inline friend GNUM operator-(GNUM x, JF y)
{
        return x - GNUM(y);
}
inline friend GNUM operator-(GNUM x, GNUM y)
{
        if (x < y)
                putchar('-'), swap(x, y);
        GNUM res = x;
        register int cnt = x.len;
        for (register int i = 1; i <= cnt; ++i)
        {
                res.a[i] -= y.a[i];
                if (res.a[i] < 0)
                        --res.a[i + 1], res.a[i] += BASE;
        }
        res.del_zero();
```

```
        return res;
    }
    template <typename CF>
    inline friend GNUM operator*(const GNUM &x, const CF &y)
    {
        return x * GNUM(y);
    }
    inline friend GNUM operator*(const GNUM &x, const GNUM &y)
    {
        GNUM res;
        res.len = x.len + y.len;
        for (register int i = 1; i <= x.len; ++i)
        {
            for (register int j = 1; j <= y.len; ++j)
            {
                res.a[i + j - 1] += x.a[i] * y.a[j];
                res.a[i + j] += res.a[i + j - 1] / BASE;
                res.a[i + j - 1] %= BASE;
            }
        }
        res.del_zero();
        return res;
    }
    template <typename GF>
    inline friend GNUM operator/(const GNUM &x, const GF &y)
    {
        GNUM res;
        res.len = x.len;
        long long k = 0;
        for (register int i = x.len; i; --i)
        {
            k = k * BASE + x.a[i];
            res.a[i] = k / y;
            k %= y;
        }
        res.del_zero();
        return res;
    }
    inline friend GNUM operator/(const GNUM &x, const GNUM &y)
    {
        GNUM res, k;
        res.len = x.len;
        for (register int i = x.len; i; --i)
        {
            k.add(x.a[i]);
            while (k >= y)
                k = k - y, res.a[i]++;
        }
        res.del_zero();
        return res;
    }
    inline bool opd()
    {
        return a[1] & 1;
    }
};
```