```
#include <iostream>
#include <bitset>
#include <vector>

template <size_t n>
class Bitset {
public:
    template<size_t n1> friend std::ostream& operator<<(std::ostream&, Bitset<n1>&);
    Bitset()
    {
        size = n;
        if (n % 16 == 0)
        {
            len = n / 32;
        }
        else
        {
            len = n / 32 + 1;
        }
        Vec = new int[len];
        for (int i = 0; i < len; i++)
        {
            Vec[i] = 0;
        }
    }
    Bitset(unsigned long long num)
    {

        size = n;
        if (n % 32 == 0)
        {
            len = n / 32;
        }
        else
        {
            len = n / 32 + 1;
        }
        Vec = new int[len];
        for (int i = 0; i < len; i++)
        {
            Vec[i] = 0;
        }

        int c = 0;
        //100 11101100 10110001
        for (int i = 0; i < size; i++)
        {
            int u = len - c - 1;
            if (u < 0)
            {
                if (num & (1 << size - 1 - i))
                {
                    Vec[c] = Vec[c] | ((num & (1 << size - 1 - i)));
                }
            }
            else
            {
                if (num & (1 << size - 1 - i))
                {
                    Vec[c] = Vec[c] | ((num & (1 << size - 1 - i)) >> 32 * u);
                }
            }
            if ((size - i) % 32 == 0 && i!=0)
            {
                c++;
            }

        }

    }
    bool all() const {
        int i = size - 32 * (len - 1) - 1;
        for (int j = 0; j < len; j++)
        {
            for (; i >= 0; i--)
            {
                if (!(Vec[j] & (1 << i))) {
                    return false;
                }
            }
            i = 31;
        }
        return true;
    }
    bool any()const {
        int i = size - 32 * (len - 1) - 1;
        for (int j = 0; j < len; j++)
        {
            for (; i >= 0; i--)
            {
                if (Vec[j] & (1 << i)) {
                    return true;
                }
            }
            i = 31;
        }
        return false;
    }
    size_t count() const {
        int c = 0;
        int i = size - 32 * (len - 1) - 1;
        for (int j = 0; j < len; j++)
        {
            for (; i >= 0; i--)
            {
                if (Vec[j] & (1 << i)) {
                    c++;
                }
            }
            i = 31;
        }
        return c;
    }
    Bitset<n>& flip() {
        int i = size - 32 * (len - 1) - 1;
        for (int j = 0; j < len; j++)
        {
            for (; i >= 0; i--)
            {
                Vec[j] = Vec[j] ^ (1 << i);
            }
            i = 31;
        }

        return *this;
    }
    Bitset<n>& flip(size_t n) {
        if (n < size)
        {
            int u = len-1-n / 32;
            int i = n % 32;
            Vec[u] = (Vec[u] ^ (1 << i));
        }
        return *this;
    }
    bool none() const {
        int i = size - 32 * (len - 1) - 1;
        for (int j = 0; j < len; j++)
        {
            for (; i >= 0; i--)
            {
                if (Vec[j] & (1 << i)) {
                    return false;
                }
            }
            i = 31;
        }
        return true;
    }
    template<size_t n2>
    Bitset<n2>& operator=(Bitset<n2>& rhs) {
        if (this == &rhs)
        {
            return *this;
        }
        delete [] this->Vec;
        size = rhs.size;
        len = rhs.len;
        Vec = new int[len];
        for (int i = 0; i < len; i++)
        {
            Vec[i] = rhs.Vec[i];
        }
        return *this;
    }
    Bitset<n>& reset() {
        for (int i = 0; i < len; i++)
        {
            Vec[i] = 0;
        }
        return *this;
    }
    Bitset<n>& reset(size_t n) {
        if (n < size)
        {
            int u = len - 1 - n / 32;
            int i = n % 32;
            if (Vec[u]&(1<<i))
            {
                Vec[u] = (Vec[u] ^ (1 << i));
            }
        }
        return *this;

    }
    Bitset<n>& set() {
        for (int i = 0; i < len; i++)
        {
            Vec[i] = 4294967295;
        }
        return *this;
    }
```

```
Bitset<n>& set(size_t n) {
    if (n < size)
    {
        int u = len - 1 - n / 32;
        int i = n % 32;
        Vec[u] = (Vec[u] | (1 << i));
    }
    return *this;

}
size_t size_() {
    return size;
}
bool test(size_t n) const {
    int u = len - 1 - n / 32;
    int i = n % 32;

     return(Vec[u] & (1 << i));

}
unsigned long to_ulong() {
    unsigned long l= 0;
    int i = size - 32 * (len - 1) - 1;
    for (int j = 0; j < len; j++)
    {
        for (; i >= 0; i--)
        {
            if ((Vec[j]&(1<<i)))
            {
                l = l | (1 << i*(len-j));
            }
        }
        i = 31;
    }
    return l;
}
unsigned long long to_ullong() {
    unsigned long long l = 0;
    int i = size - 32 * (len - 1) - 1;
    for (int j = 0; j < len; j++)
    {
        for (; i >= 0; i--)
        {
            if ((Vec[j] & (1 << i)))
            {
                l = l | (1 << i * (len - j));
            }
        }
```

```
            i = 31;
        }
        return l;
    }
    std::string to_string() {
        std::string s = "";
        int i = size - 32 * (len - 1) - 1;
        for (int j = 0; j < len; j++)
        {
            for (; i >= 0; i--)
            {
                if ((Vec[j] &(1<<i)))
                {
                    s+='1';
                }
                else
                {
                    s+='0';
                }
            }
            i = 31;
        }

        return s;
    }
    std::string to_string(char c) {
        std::string s = "";
        int i = size - 32 * (len - 1) - 1;
        for (int j = 0; j < len; j++)
        {
            for (; i >= 0; i--)
            {
                if ((Vec[j] & (1 << i)))
                {
                    s += '1';
                }
                else
                {
                    s += c;
                }
            }
            i = 31;
        }

        return s;
    }
    std::string to_string(char c1, char c2) {
        std::string s = "";
        int i = size - 32 * (len - 1) - 1;
        for (int j = 0; j < len; j++)
        {
            for (; i >= 0; i--)
            {
                if ((Vec[j] & (1 << i)))
                {
                    s += c2;
                }
                else
                {
                    s += c1;
                }
            }
            i = 31;
        }

        return s;
    }

private:
    size_t size;
    int* Vec;
    int len;
};
template<size_t n>
std::ostream& operator<<(std::ostream& o, Bitset<n>& b) {
    int i = b.size - 32 * (b.len - 1) - 1;
    for (int j = 0; j < b.len; j++)
    {
        for (; i >= 0; i--)
        {
            o << (bool)(b.Vec[j] & (1 << i));
        }
        i = 31;
        o << ' ';
    }
    o << '\n';
    return o;
}



int main() {
    Bitset<32> b(78657884);
    std::cout << b<<b.to_ulong();

}
```