Lab 6: Array of bits

[deadline**: 20th May 2022]**

Prologue

Information theory is the scientific study of the quantification, storage, and communication of information. The field was fundamentally established by the works of Harry Nyquist and Ralph Hartley in the 1920s, and Claude Shannon in the 1940s. The field is at the intersection of probability theory, statistics, computer science, statistical mechanics, information engineering, and electrical engineering.

The *bit* is a basic unit of information in computing and digital communications. The bit represents a logical state with one of two possible values. These values are most commonly represented as either 1 or 0, but other representations such as true/false, yes/no, +/−, or on/off are common..

Task

Define the class `bitarray` for representing an array of bits. The array of bits should be designed in such way that all the bits allocated are used (modulo word size). In the `bitarray` class, define an index operator that would allow both reading from and writing single bits to it. Here's a piece of code that should compile and run:

```
bitarray t(53); // array of 53 bits (array initialized with 0)
bitarray u(37ull); // array of 64 bits (sizeof(uint64_t)*8)
bitarray v(t); // array of 53 bits (copy constructor)
bitarray w(bitarray(8){1, 0, 1, 1, 0, 0, 0, 1});
                                  // array of 8 bits (move constructor)
v[0] = 1; // assignment 1
t[43] = false; // assignment 0
bool b = v[1]; // read v[1]
u[17] = u[29] = u[63]; // cascading assignment
cout << t << endl; // writing on console
```

You cannot address a single bit (and thus you cannot establish a reference to it), so you have to use a special technique to access a single bit in the array. It is enough to use objects of the `ref` auxiliary class, which can read and write a single bit in the array.

```
class bitarray {
    typedef uint64_t word; // one cel in array
    static const int wordsize; // word size = 64 bits
    friend istream & operator >> (istream &we, bitarray &arr);
    friend ostream & operator << (ostream &wy, const bitarray &arr);
    class ref; // auxiliary class
protected:
    int len; // numer of bits
    word *arr; // array of bits
public:
    explicit bitarray(int siz);
    explicit bitarray(word w);
    bitarray(const bitarray &arr); // copy constructor
    bitarray(bitarray &&arr); // move constructor
    bitarray& operator= (const bitarray &arr); // copy assignment
    bitarray& operator= (bitarray &&arr); // move assignment
    ~bitarray(); // destructor
private:
    bool read(int i) const; // read single bit
    bool write(int i, bool b); // write single bit
public:
    bool operator[] (int i) const; // index operator
    ref operator[] (int i); // index operator
    inline int size() const; // size of array
public:
    // bitwise operators: |, |=, &, &=, ^, ^=, !
};
```

Complete the definition of the `bitarray` class. The `ref` class is an auxiliary class that is to address a single bit in the array. Place the class `bitarray` in the namespace `data_structures`.

Finally write a program, which reliably tests the `bitarray` class. All objects in your program should be created on the heap (use operator `new`).

Whenever you encounter any errors, ambiguities or contradictions in the program, this should be signaled by an exception.