1. Sa se implementeze clasa urmatoare ce reprezinta numere mari si operatii cu acestea:

```
class BigDecimal{
                byte *cifre;
                int len;
        public:
                BigDecimal(byte* a, int lungime);
                BigDecimal(BigDecimal&);
                BigDecimal(long number);

                ~BigDecimal();

                friend istream& operator >> (istream&, BigDecimal&);
                friend ostream& operator<< (ostream&, BigDecimal&);
                BigDecimal operator+ (BigDecimal&);
                int operator < (BigDecimal&);
};
```

2. Sa se implementeze clasa urmatoare ce reprezinta siruri de caractere si operatii cu acestea:

```
class String{
                char *p;
                int len;
        public:
                String(char* mp);
                String(String& s);

                ~String();

                String& operator= (String& s);
                String    operator+ (String& s);
                friend ostream& operator << (ostream& o, String& s);
                String operator- (String& s); //elimina toate aparitiile sirului s.
};
```

3. Sa se implementeze clasa urmatoare ce reprezinta tipul multime si operatii cu acesta:

```
class Multime{
                int* elem;
                int number;
        public:
                Multime(int m_number, int* m_elem);
                Multime(Multime&);

                ~Multime();

                Multime& operator= (Multime& m);
                Multime operator+ (Multime& m);
                Multime operator* (Multime& m);
                Multime operator- (Multime& m);
                Friend ostream& operator << (ostream& o, Multime& m);
};
```

4. Sa se implementeze clasa urmatoare ce reprezinta polinoame si operatii cu acestea:

```
#define MAX 20

class Polinom{
                float coef[MAX];
                int grad;
        public:
                Polinom(int m_grad, int* m_coef);
                Polinom(Polinom&);

                friend ostream& << (ostream& o, Polinom& p);
                Polinom operator + (Polinom& p);
                Polinom operator * (Polinom& p);
};
```

5. Sa se implementeze clasa urmatoare ce reprezinta matrici si operatii cu acestea:

```
class Linie{
                int* p;
        public:
                Linie(int nrColoane);

                ~Linie();

                int& operator[] (int coloana);
};

class Matrice{
                Linie* pMatrice;
                Int nrLinii, nrColoane;
        public:
                Matrice(int, int);
                Matrice(Matrice& m);

                ~Matrice();

                int getLinii();
                int getColoane();
                Matrice operator* (Matrice& m);

                friend ostream operator << (ostream& o, Matrice& m);
                Linie& operator[] (int index);
};
```

6. Sa se implementeze clasa urmatoare ce reprezinta numere rationale si operatii cu acestea:

```
class Rational{
                int numitor, numarator;
                float nrReal;
        public:
```

```
        Rational(int _numitor, int _numarator);
        Rational(Rational&);
        Rational(int _numarator);

        friend ostream& operator<<(ostream& , Rational& );
        Rational operator+ (Rational& );
        Rational& operator= (Rational& );
        Rational operator~ ();
        int operator < (Rational& );
        int operator == (Rational&);
        friend Rational operator* (Rational& );
};
```

7. Sa se implementeze clasa urmatoare ce reprezinta tipul vector si operatii cu acestea:

```
class Vector{
        void **a;
        int capacity;
        int count;
    public:
        Vector(Vector&);
        Vector(int _capacity);

        ~Vector();

        void addElement(void *);
        void insertElement(void *, int poz);
        void removeElement(void *);
        void removeAllElements();
        void* operator[] (int index);
        friend ostream& operator<< (ostream& , Vector& );
};
```

8. Sa se implementeze o structura de tip stiva:

```
class Stiva{
        void ** tab;
        int head;
        int capacity;
    public:
        Stiva();
        Stiva(int _capacity);
        Stiva(Stiva& );

        ~Stiva();

        void* pop();
        void* peek();
        void push(void*);

        int isEmpty();
        int isFull();
};
```

9. Sa se implementeze o structura de tip coada circulara.

```
class Queue{
            void** tab;
            int first, last;
            int capacity;
      public:
            Queue();
            Queue(int _capacity);
            Queue(Queue& );

            ~Queue();

            void* deQueue();
            void enQueue(void*);

            int isEmpty();
            int isFull();
};
```

10. Implementati o lista simplu inlantuita si efectuati operatii ca: inserarea unui nod la inceput, la sfarsit si dupa un anumit element, stergerea unui nod, concatenarea a doua liste.

11. Sa se defineasca clasa Complex utila in realizarea unor operatii cu numere complexe.

12. Sa se implementeze clasa ArboreBinar si sa se implementeze toate metodele de parcurgere a unui arbore binar.