Enunturi probleme

1. Se consideră un program care descrie organizarea personalului unei instituții folosid claselor derivate. O clasă numită `Angajat` deține date şi funcții referitoare la un angajat al instituției:

```
class Angajat{
      char * nume;
      float salariu;
public:
      Angajat();
      Angajat(char *n, int s, float sal);
      Angajat(Angajat& r);
      void display();
      float getSalariu();
      void setSalariu(float s);
};
```

Diferite categorii de angajați necesită date suplimentare față de cele definite în clasa `Angajat`, corespunzătoare postului pe care îl dețin. De exemplu, un şef de secție (administator) este un angajat (deci sunt necesare toate datele care descriu această calitate) dar mai sunt necesare şi alte informații, de exemplu precizare secției pe care o conduce.

Un `administrator` este un `angajat`, de aceea clasa `Administrator` se poate construi prin derivare din clasa `Angajat` astfel:

```
class Administrator : public Angajat {
      int sectie;
public:
      Administrator(const char *n, float sal, int sec);
      Administrator(Administrator& r);
      ~Administrator();
};
```

Sa se implementeze aceasta ierarhie de clase.

2. Sa se implementeze un vector(tablou) in care se pot pastra obiecte de tipuri diferite definite de utilizator. Pentru aceasta se defineşte clasa `Object` ca o clasă de bază pentru toate tipurile derivate:

```
class Object{
public:
      Object() {};
      virtual ~Object() {};
      virtual void display() = 0;
};
```

Clasa `ObArray`, defineşte un vector de pointeri de tip `Object*` . Nu este necesar să fie limitată dimensiunea vectorului deoarece se asigură creşterea dimensiunii acestuia atunci când este necesar.

```
class ObArray : public Object {
      Object **p;       // vector de pointeri
      int size;         // numar de elemente la un moment dat
      int grows;        // increment de creştere a dimensiunii
      int dimens;       // dimensiune vector
public:
```

```
        ObArray(int size=0,int grows, int dimens);//Constructor
        ~ObArray();//Destructor
        void RemoveAll();//Elimina toate obiectele din vector
        int getSize(); // Intoarce numarul de elemente din vector.
        int add(Object* x); //Adauga un element la vector.
        int insertAt(int i, Object *x);//Insereaza un element pe
                                       // o pozitie data
        int removeAt(int i); //Elimina elementul de pe pozitia i
        Object* getAt(int i); //Intoarce elementul de la pozitia i
        void display();// Afisare elementelor din tablou.
};
```

Sa se utilizeze aceste clase pentru a memora un tablou de Puncte si un tablou de numere Complexe.

3. Sa se implementeze o ierarhie de clase pentru a realiza operatii cu multimi ale caror elemente vor fi de tipuri diferite.

```
class Element:
      public:
            virtual ~Element();
            virtual void display() = 0;
            virtual void read() = 0;
            int equals(Element&) = 0;
            virtual char *getClassName()=0;
};

class Set{
      private:
            int size;
            Element *elements;
      public:
            Set(int size);
            Set(int size,Element *pe);
            Set(Set&);
            ~Set();
            void insert(Element*);
            void remove(Element*);
            int lookup(Element*); // Cautarea unui element
            Set operator +(Set &);// reuniunea
            Set operator –(Set &);// diferenta
            Set operator *(Set &);// intersectia
            int operator ==(Set &);// Testarea egalitatii
            int operator <(Set &);// Tesstarea incluziunii
            void read();//Citirea elementelor multimii
            void display();//Afisarea elementelor multimii.
};
```

Sa se realizeze utilizand aceste clase intersectia a 2 multimi ale caror elemente sunt caractere.

4. Sa se implementeze o clasa care sa retina revistele si cartile dintro biblioteca. Pentru aceasta se va folosi o lista sipmlu inlantuita ale carei elemente sunt revistele si/sau cartile din biblioteca.

```
//Clasa de baza pentru clasa Revista si Carte
class Articol{
   protected:
      int cota;//Cota cartii sau revistei
      char *titlul;
   public:
```

```
      Articol(int cota, char *titlul);
      Articol(Articol&);
      virtual ~Articol();
      int getCota();
      void setCota(int);
      char *getTitlul();
      void setTitlul();
      virtual void display();
      virtual void read();
}

class Revista: public Articol{
      int nr;//Numarul revistei
      int tiraj;
      int frecv;// Numarul de aparitii pe luna
  public:
      Revista(int cota,int titlul,int nr,int tiraj,int frecv);
      Revista(Revista&);
      ~Revista();
      void display();
      void read();
};

class Carte: public Articol{
      char *autor;
      char *editura;
      int an;//anul aparitiei
  public:
      Carte(int cota,int titlul,char * autor,
             char *editura, int an);
      Carte(Carte&);
      ~Carte();
      void display();
      void read();
};

class Nod {
      Articol *info;
      Nod *next;
  public:
      Nod(Articol *,Nod *next=0);
      Articol* getInfo();
      void setInfo(Articol *);
      Nod *getNext();
      Void setNext(Nod*);
};

class Biblioteca{
      Nod *prim;
  public:
      Bibioteca();
      void addArticol(Articol *);
      int removeArticol(Articol *);
      Articol * find(int cota);
      Articol * find (char *titlul);
      void display();

}
```

5. Se considera clasa `Abonat` care deriva in clasa `Persoana`. Supraincarcati operatorii <<, >>. Sa se construiasca clasa `Agenda` ce contina o lista de abonati si sa se supraincarce operatorul [ ](indexare)care returneaza abonatul cu numele precizat .

6. Sa se implementeze clasa `Telefon` (nr_tel,nume_operator). Sa se deriveze din aceasta clasa `TelefonMobil`(autonomie, tip_acumulator).

7. Sa se implementeze clasa abstracta `FiguraGeometrica`, din care sa se derive `Cerc` si `Poligon`. Sa se deriveze din `Poligon` clasa `Patrulater` si `Triunghi`. Din `Patrulater` sa se deriveze clasa `Dreptunghi`.
Functii virtuale : `getAria,getPerimetru`

8.Implementati urmatoarea ierarhie de clase:

```
class Localitate{
               protected:
                   char *denumire;
                   int cod;
                   long nr_locuitori;
               public:
                  ……
            virtual void display();
};
class  Oras :public Localitate{
                     protected :
                         int nrBlocuri;
                      public:
                      ……..
};
class Capitala :public Oras{
                      protected:
                         char *numePrefect;
                       public:
                       ……
};
class Judet {
                   Localitate *p;
                   int nrLoc;
               public:
                   …….
};
```

9. Realizati un program care citeste dintr-un fisier text *n* numere complexe si calculeza produsul lor.

10. Sa se implemente o clasa `Grupa` ce retine studentii dintr-o grupa. Clasa va avea metode pentru salvarea repectiv incarcarea grupei intr-un/dintr-un fisier.

11. Sa se realizeze un program C++ ce realizeaza concatenarea mai multor fisiere text al caror nume se preiau in linia de comanda. E vor utiliza stream-uri.

12. Sa se implementeze clasa `Matrice` utilizand template-uri reprezentata sub forma liniara ale carei elemente pot fi numere intregi, reale. Sa se supraincarce operatorii : +,-,*,citire (>>),afisare (<<) .

13. Sa se implementeze clasa `Multime` ale carei elemente pot fi de orice tip. Supraincarcati operatorii: +(reuniune), -(diferenta), *(intersectie).

14. Implementati clasa `ArboreBinar` ce implementeaza notiunea de arbore binar ale carului noduri pot fi de orice tip: numere intregi,reale, obiecte. Clasa va contine metodele:

- creare arbore
- parcurgere in inordine
- parcurgere in postordine
- parcurgere in preordine

15. Implementati clasa `ArboreOarecare` ce implementeaza notiunea de arbore oreacare ale carului noduri pot fi de orice tip: numere intregi, reale, obiecte. Clasa va contine metodele:

- creare arbore
- parcurgere in a-preordine
- parcurgere in a-postordine

16. Sa se implementeze clasa Coada ale carei element pot fi de orice tip.