INSTITUTO POLITÉCNICO DE BRAGANÇA
Escola Superior de Tecnologia e Gestão

Engenharia Informática
Informática de Gestão
Época Normal – 2 de fevereiro de 2018

Grupo I (15 valores)

a) Defina em C++ todas as classes do problema, respeitando integralmente os métodos e ... [11 val.]

```
-44-
class Condutor{      -6-                                                    1
    string carta, nome;
    Viatura *viatura;                                                       1
public:
    Condutor(string cart, string nom) :carta(cart), nome(nom){viatura=NULL;} 1
    Viatura *getViatura(){return viatura;}                                  1
    void setViatura(Viatura *v){viatura=v;}                                 1
    bool operator<(const Condutor &outro)const { return carta < outro.carta;}1
};

class Viatura{       -6-                                                    1
    string matricula;
    Condutor *condutor;                                                     1
public:
    Viatura(string mat) :matricula(mat){ condutor = NULL; }                 1
    Condutor *getCondutor()const { return condutor; }                       1
    void setCondutor(Condutor *c){ condutor=c;}                             1
    bool operator<(const Viatura &out)const{return matricula<out.matricula;} 1

};


class Autoaluga{     -33-
    Colecao<Viatura> viaturas;                                              2
    Colecao<Condutor> condutores;                                           2

    Viatura *findViatura(string mat) {                                      2
        Viatura v(mat);
        return viaturas.find(v);
    }

    Condutor *findCondutor(string cart) {                                   2
        Condutor c(cart, "");
        return condutores.find(c);
    }

public:
    bool addCondutor(string cart, string nom){                              2
        Condutor c(cart,nom);
        return condutores.insert(c);
    }
```

```
    bool addViatura(string mat){                                              2
        Viatura v(mat);
        return viaturas.insert(v);
    }

    bool levantarViatura(string cart, string mat){                            8
        Condutor *c = findCondutor(cart);
        if (c == NULL) return false; //Condutor inexistente!
        else
            if (c->getViatura() != NULL) return false; //Conduz outra viatura!
            else{
                Viatura *v = findViatura(mat);
                if (v == NULL) return false; //Viatura inexistente!
                else
                if (v->getCondutor() != NULL) return false; //V. já com condutor!
                    else{
                        c->setViatura(v);
                        v->setCondutor(c);
                        return true;
                    }
            }
    }

    bool devolverViatura(string mat){                                         6
        Viatura *v = findViatura(mat);
        if (v == NULL) return false; //Viatura inexistente!
        else
            if (v->getCondutor() == NULL) return false; //V. não alugada
            else{
                v->getCondutor()->setViatura(NULL);
                v->setCondutor(NULL);
                return true;
            }
    }

    bool removerViatura(string mat) {                                         6
        Viatura *v = findViatura(mat);
        if (v != NULL && v->getCondutor() == NULL){//Existe e não está alugada
            viaturas.erase(Viatura(mat));
            return true;
        }
        else return false;
    }

};
```

b) Acrescente ao problema os métodos que permitam apresentar na saída… [4 val.]

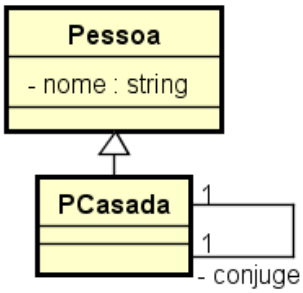
```
-16-
void Condutor::print() const{cout << nome << " - carta " << carta;}          3

string Viatura::geMatricula()const { return matricula; }                    2

void Autoaluga::mostrarViaturasAlugadas() const{                            1
    Colecao<Viatura>::iterator it;                                          1
    for(it=viaturas.begin(); it!=viaturas.end(); it++)                      2
        if (it->getCondutor() != NULL) {                                    3
            cout << "Viatura " << it->geMatricula() << " conduzida por ";   1
            it->getCondutor()->print();                                     3
            cout << endl;
        }
}
```

Grupo II (5 valores)

a) Desenhe o diagrama de classes UML do problema… [1 val.]



b) Indique três métodos (construtores ou não) que, embora não definidos explicitamente. [.9 val.]

Construtor de cópia, destrutor e operador afetação.

c) Apresente o resultado que será visualizado na saída standard após a execução do programa. [2.4 val.]

```
-0-
Criado Manuel
Criado Maria
Conjuge 1
Conjuge 2
-1-
Manuel + Maria
-2-
Criado Ana
Criado Luis
Conjuge 1
Conjuge 2
-3-
Ana + Luis
-4-
Manuel + Ana
-5-
Ana + Manuel
```

d) Apresente o resultado que seria visualizado na saída standard se nenhum dos métodos. [0.7 val.]

O mesmo.