INSTITUTO POLITÉCNICO DE BRAGANÇA
Escola Superior de Tecnologia e Gestão

Engenharia Informática
Informática de Gestão
Época de Recurso – 10 de fevereiro de 2020

Grupo I (15 valores)

a) Defina em C++ todas as classes do problema, respeitando integralmente os métodos e... [9.9 val.]

```
Class MarcViagens {                                                    15
      Colecao<Cliente> clientes;
      Colecao<Voo> voos;                                    3

      Cliente *findCliente(int n) const{                    2
            Cliente c(n, "");
            return clientes.find(c);
      }
      Voo *findVoo(int n) const{                            2
            Voo v(n,"",TDataHora(),0);
            return voos.find(v);
      }
public:
      bool addCliente(int num, string nom) {                2
            Cliente c(num, nom);
            return clientes.insert(c);
      }
      bool addVoo(int num, string dest, TDataHora dh, int nlugs) {  2
            Voo v(num, dest, dh, nlugs);
            return voos.insert(v);
      }
      bool marcarViagem(int cliente, int voo, int lugar)    4
            Cliente *c = findCliente(cliente);
            if (c == NULL) return false; //Cliente inexistente!
            else {
                  Voo *v = findVoo(voo);
                  if (v == NULL) return false; //Voo inexistente!
                  else return v->addReserva(c, lugar);
            }
      }

};

class Reserva {                                                        4
      int lugar;              2
      Cliente *cliente;
      Voo *voo;
public:
      Reserva(int lug, Cliente *c, Voo *v) {      1
            lugar = lug;
            cliente = c;
            voo = v;
      }
      bool operator<(const Reserva &outro)const {return lugar < outro.lugar;}  1
};
```

```
class Cliente{                                                          6
      int num;
      string nome;
      Colecao<Reserva*> reservas;      2
public:
      Cliente(int nu, string no) { num = nu; nome = no; }      1
      bool add(Reserva *r) { return reservas.insert(r); }      2
      bool operator<(const Cliente &outro)const { return num < outro.num; }  1

};

class Voo{                                                              8
      int num;              2
      string destino;
      TDataHora datahora;
      int nlugares;
      Colecao<Reserva> reservas;
public:
      Voo(int n, string d, TDataHora dh, int nl){          1
            num = n; destino = d; datahora = dh; nlugares = nl;
      }
      bool addReserva(Cliente *c, int lug) {          4
            if (lug<1 || lug>nlugares) return false;
            Reserva r(lug, c, this);
            if (reservas.insert(r))
                  return c->add(reservas.find(r));
            else return false;
      }
      bool operator<(const Voo &outro)const { return num < outro.num;}    1
};
```

b) Acrescente ao problema os métodos que permitam apresentar na saída standard. [2.1 val.]

```
void MarcViagens::mostrarVoosLotados() const {          4                7
      Colecao<Voo>::iterator it;
      for (it = voos.begin(); it != voos.end(); it++)
            if (it->lotado()) cout<<"Voo " << it->getNumero() << " lotado." << endl;
}

bool Voo::lotado() const {                    2
            return reservas.size() == nlugares;
}

int Voo::getNumero() const {          1
            return num;
}
```

c) Acrescente ao problema os métodos que permitam cancelar uma reserva... [3 val.]

```
bool MarcViagens::cancelarReserva(int voo, int lugar) {      3          10
      Voo *v = findVoo(voo);
      if (v == NULL) return false; //Voo inexistente!
      else return v->cancelarReserva(lugar);
}
```

```
bool Voo::cancelarReserva(int lugar) {                    4
      Reserva r(lugar, NULL, NULL);
      Reserva *pr = reservas.find(r);
      if (pr == NULL) return false;
      else {
      pr->getCliente()->cancelarReserva(pr);
            reservas.erase(r);
            return true;
      }
}

*Reserva::getCliente()const { return cliente; }           1

void Cliente::cancelarReserva(Reserva *r) { return reservas.erase(r); } 2
```

Grupo II (5 valores)

a) Diga quais são os métodos construtores (definidos explicitamente ou não) que estão ... [0.6 val.]
Nas classes Epoca e EpRecurso: construtor de cópia e construtor de conversão;
Na classe EpNormal: construtor de cópia e o construtor de 3 parâmetros EpNormal(double,double,double).

b) Implemente o método nota() da classe EpNormal de forma a devolver o valor da nota... [0.9 val.]

```
int nota()const {
      if (Epoca::nota() < notaMinimaExame) return Epoca::nota();
      else {
            double val = miniteste * 0.3 + trabalho * 0.2 + Epoca::nota()*0.5;
            return (int)(val + 0.5);
      }
}
```

c) Apresente o resultado que será visualizado na saída standard após a execução do programa. [1.5 val.]

```
Inicio de Epoca...
Inicio da Epoca Normal...
Inicio de Epoca...
Inicio da Epoca de Recurso...
-1-
-2-
Epoca Normal: aprovado com 13 valores.
Epoca de Recurso: aprovado com 11 valores.
Classificacao final: 13
-3-
Fim da Epoca de Recurso
Fim de Epoca:
Fim da Epoca Normal
Fim de Epoca:
```

d) Apresente o resultado que será visualizado na saída standard após a execução do programa. [0.8 val.]

```
-2-
Epoca Normal: aprovado com 13 valores.
Epoca de Recurso: aprovado com 11 valores.
Classificacao final: 13
-3-
```

e) Diga que consequências é que teria, naquilo que é visualizado, o destrutor .... [0.4 val.]
Nada se alterava.

f) Por que razão numa coleção híbrida são normalmente guardados os apontadores para .... [0.8 val.]
Só assim, tirando-se partido da conversão ascendente (*upcast*), se consegue misturar na coleção elementos de diferentes tipos.
Explicando melhor: na criação da estrutura de dados que define uma coleção híbrida, começa-se por declarar que a mesma coleciona apontadores para uma dada classe base, para depois guardarem-se nela apontadores para objetos de diferentes classes de si derivadas.