Parte I. Considere o seguinte programa:

```
 1 #include <iostream>
 2 using namespace std;
 3 void soma3(int X, int &Y)
 4 { X = X + 5;
 5   Y = Y + 5;
 6 }
 7 int main()
 8 { int A, B;
 9   A = 0;
10   B = 0;
11   soma3(A,B);
12   cout << A << " " << B << endl;
13   soma3(B,A);
14   cout << A << " " << B << endl;
15   return 0;
16 }
```

a) Indique a saída resultante da execução do programa. Explique como o resultado foi obtido. 0 , 5 5 , 5

b) No programa, substitua a linha (11) pelo comando:

(i) `soma3(A,A);` 5 , 0 10 , 0

(ii) `soma3(0,A);` 5 , 0 10 , 0

(iii) `soma3(A,0);` erro de tipo de parâmetro

Tente executar. O que acontece em cada caso? Por quê?

Parte II. Considere o seguinte programa:

```
 1 #include <iostream>
 2 using namespace std;
 3 //------------------------
 4 int soma1(int a, int b)
 5 { b = a + b;
 6   return b;
 7 }
 8 //------------------------
 9 int soma2(int &a, int b)
10 { b = a + b;
11   return b;
12 }
13 //------------------------
14 int soma3(int a, int &b)
15 { b = a + b;
16   return b;
17 }
18 //------------------------
19 int main()
20 { int x=3, y=4;
21   cout << "Valores iniciais x=" << x << " e y=" << y << endl;
22   cout << "soma1(" << x << "," << y << ")" << endl;
23   cout << soma1(x,y) << endl;
24   cout << "soma2(" << x << "," << y << ")" << endl;
25   cout << soma2(x,y) << endl;
26   cout << "soma3(" << x << "," << y << ")" << endl;
27   cout << soma3(x,y) << endl;
28   cout << "Valores finais x=" << x << " e y=" << y << endl;
29   return 0;
30 }
```

a copia de b recebe:
copia de a + copia de b

a copia de b recebe:
o valor de a + opia de b

a variável b recebe:
a copia de a + o valor de b

a) Indique o(s) tipo(s) de passagem de parâmetros das funções, `soma1`, `soma2` e `soma3`;

b) Qual a saída resultante da execução do programa, ou seja, tudo o que é escrito na tela?

x=3 y=4
7
7
7
x=3 y=7

Parte III. Podemos representar tipos de dados por meio de estruturas (registros). Por exemplo, um tipo de dados capaz de armazenar o tempo pode ser representada por três componentes (ou campos): hora, minuto e segundo. Isso pode ser declarado como:

```
// Definicao do tipo abstrato de dados Time
struct Time
{  int hour;                    // 0 - 23
   int minute;                  // 0 - 59
   int second;                  // 0 - 59
};
```

Uma vez definida uma nova estrutura de dados ela pode ser utilizada no programa para declarar variáveis do novo tipo, por exemplo, o comando:

```
Time a,b;
```

Declara duas variáveis, `a` e `b`, do tipo de dados `Time`. Para se ter acesso aos campos, segue-se a convenção

<nome-da-variável>.<nome-do-campo>

Assim, para atribuir o valor 12 ao campo `hour` da variável `a`, basta o seguinte comando:

```
a.hour = 12;
```

Estruturas de um mesmo tipo podem ser atribuídas entre si. Por exemplo, após os comandos:

```
a.hour   = 12;
a.minute = 10;
a.second = 15;
b = a;
```

o conteúdo da variável `b` recebe uma cópia do conteúdo da variável `a`, campo a campo. Note que o fragmento de código anterior é equivalente ao seguinte código:

```
a.hour   = 12;
a.minute = 10;
a.second = 15;
b.hour   = a.hour;
b.minute = a.minute;
b.second = a.second;
```

Com base nessas informações, indique a saída resultante da execução do seguinte programa:

```
 1  #include <iostream>
 2  using namespace std;
 3  //------------------------
 4  // Definicao do tipo abstrato de dados Time
 5  struct Time
 6  {  int hour;                    // 0 - 23
 7     int minute;                  // 0 - 59
 8     int second;                  // 0 - 59
 9  };
10  //------------------------
11  void p1(Time t)
12  { t.hour = 0;
13    t.minute = 0;          zera a cópia da estrutura
14    t.second = 0;
15  }
16  //------------------------
17  void p2(Time &t)
18  { t.hour = 0;
19    t.minute = 0;          zera a estrutura
20    t.second = 0;
21  }
22  //------------------------
```

```
23  int main()
24  {  Time almoco;
25
26     // atribuir valores validos
27     almoco.hour = 12;
28     almoco.minute = 30;
29     almoco.second = 15;
30     cout << "Inicialmente, almoco sera servido as ";
31     cout << almoco.hour << ":" << almoco.minute << ":" << almoco.second << endl;
32                                        12:30:15
33     p1(almoco);
34     cout << "Apos p1, almoco sera servido as ";
35     cout << almoco.hour << ":" << almoco.minute << ":" << almoco.second << endl;
36                                          12:30:15
37     p2(almoco);
38     cout << "Apos p2, o almoco sera servido as ";
39     cout << almoco.hour << ":" << almoco.minute << ":" << almoco.second << endl;
40
41     return 0;                           0:0:0
42  }
```