Parte I. Seja o tipo abstrato de dados **Time** definido a seguir usando **struct**.

a. Implemente-o, informando qual a saída resultante da sua execução.
b. Explique o que acontece ao se atribuir valores inválidos aos campos dessa estrutura.

```
 1 // time.cpp
 2 // Cria uma estrutura, seus elementos de dados e imprime-a.
 3 #include <iostream>
 4 using namespace std;
 5
 6 // Definicao do tipo abstrato de dados Time
 7 struct Time
 8 {  int hour;                     // 0 - 23
 9    int minute;                   // 0 - 59
10    int second;                   // 0 - 59
11 };
12
13 void printMilitary(Time t)
14 // Pre: Horario valido t seja fornecido.
15 // Pos: Horario no formato 24 horas e' impresso.
16 {
17    cout << (t.hour   < 10 ? "0" : "") << t.hour << ":"
18         << (t.minute < 10 ? "0" : "") << t.minute << ":"
19         << (t.second < 10 ? "0" : "") << t.second;
20 }
21
22 void printStandard(Time t)
23 // Pre: Horario valido t seja fornecido.
24 // Pos: Horario no formato 12 horas (AM/PM) e' impresso.
25 {
26    cout << ((t.hour == 0 || t.hour == 12) ? 12 : t.hour % 12)
27         << ":" << (t.minute < 10 ? "0" : "") << t.minute
28         << ":" << (t.second < 10 ? "0" : "") << t.second
29         << (t.hour < 12 ? " AM" : " PM");
30 }
31
32 int main()
33 {  Time dinnerTime;        // variavel do novo tipo Time
34
35    // atribuir valores validos
36    dinnerTime.hour = 18;
37    dinnerTime.minute = 30;
38    dinnerTime.second = 0;
39    cout << "Jantar sera servido as ";
40    printMilitary(dinnerTime);
41    cout << " horario militar, \nque corresponde a ";
42    printStandard(dinnerTime);
43    cout << " horario padrao.\n\n";
44
45    // atribuir valores invalidos
46    dinnerTime.hour = 29;
47    dinnerTime.minute = 73;
48    cout << "Horario com valores invalidos ";
49    printMilitary(dinnerTime);
50    cout << " horario militar, \nque corresponde a ";
51    printStandard(dinnerTime);
52    cout << " horario padrao.\n\n";
53    return 0;
54 }
```

Parte II. Na Parte I vimos que é possível atribuir valores inconsistentes a uma estrutura de dados (**struct**). Considere a implementação do tipo abstrato de dados **Time** definido a seguir usando **class**.

c. Implemente-o, informando qual a saída resultante da sua execução.
d. Explique o que acontece ao se atribuir valores inválidos aos campos dessa estrutura (linhas 25-30 do arquivo **`dtime1.cpp`**).

```
 1 // time1.h
 2 // Declaracao do objeto Time.
 3 // Métodos definidos em time.cpp
 4
 5 // Não incluir header multiplas vezes
 6 #ifndef TIME1_H
 7 #define TIME1_H
 8
 9 // Definicao do tipo abstrato de dados Time
10 class Time
11 { public:
12    Time();                          // construtor default
13    void setTime(int, int, int);     // set hour, minute e second
14    void printMilitary();            // imprime hora em formato militar
15    void printStandard();            // imprime hora em formato padrao
16   private:
17    int hour;                        // 0 - 23
18    int minute;                      // 0 - 59
19    int second;                      // 0 - 59
20 };
21
22 #endif
```

```
 1 // time1.cpp
 2 // Definicao dos Metodos para objeto Time.
 3 #include <iostream>
 4 #include "time1.h"
 5 using namespace std;
 6
 7 Time::Time()
 8 // Pre: Nenhuma
 9 // Pos: Construtor Time inicia cada elemento de dados em zero,
10 //      assegurando que todos os objetos Time iniciem num estado consistente.
11 {
12    hour = minute = second = 0;
13 }
14
15 void Time::setTime(int h, int m, int s)
16 // Pre: Valores validos para hora (h), minuto (m) e segundo (s) sejam fornecidos.
17 // Pos: Atribui novos valores de hora (24 horas).
18 //      Verificacao da validade dos valores fornecidos.
19 //      Atribui zero aos valores invalidos (estado consistente).
20 {
21    hour   = (h >= 0 && h < 24) ? h : 0;
22    minute = (m >= 0 && m < 60) ? m : 0;
23    second = (s >= 0 && s < 60) ? s : 0;
24 }
25
26 void Time::printMilitary()
27 // Pre: Nenhuma.
28 // Pos: Horario no formato 24 horas e' impresso.
29 {
30    cout << (hour < 10 ? "0" : "") << hour << ":"
31         << (minute < 10 ? "0" : "") << minute << ":"
32         << (second < 10 ? "0" : "") << second;
33 }
34
35 void Time::printStandard()
36 // Pre: Nenhuma.
37 // Pos: Horario no formato 12 horas (AM/PM) e' impresso.
38 {
39    cout << ((hour == 0 || hour == 12) ? 12 : hour % 12)
```

```
40          << ":" << (minute < 10 ? "0" : "") << minute
41          << ":" << (second < 10 ? "0" : "") << second
42          << (hour < 12 ? " AM" : " PM");
43  }
```

```
1  // dtime1.cpp
2  // Driver para testar novo tipo de dados Time
3  #include <iostream>
4  #include "time1.h"
5  using namespace std;
6
7  int main()
8  {  Time dinnerTime;              // instanciar objeto dinnerTime da classe Time
9
10    cout << "O horario inicial e ";
11    dinnerTime.printMilitary();
12    cout << " horario militar, \nque corresponde a ";
13    dinnerTime.printStandard();
14    cout << " horario padrao.\n\n";
15
16    // atribuir valores validos
17    dinnerTime.setTime(18,30,0);
18    cout << "Jantar sera servido as ";
19    dinnerTime.printMilitary();
20    cout << " horario militar, \nque corresponde a ";
21    dinnerTime.printStandard();
22    cout << " horario padrao.\n\n";
23
24    // tentativa de atribuir valores invalidos
25    dinnerTime.setTime(99,99,99);
26    cout << "Apos tentativa de atribuir valores invalidos ";
27    dinnerTime.printMilitary();
28    cout << " horario militar, \nque corresponde a ";
29    dinnerTime.printStandard();
30    cout << " horario padrao.\n\n";
31    return 0;
32 }
```

e. Explique o que acontece com o seguinte programa ao tentar compilá-lo.

```
1  // dtime2.cpp
2  // Driver para mostrar acesso aos elementos do tipo de dados Time
3  #include <iostream>
4  #include "time1.h"
5  using namespace std;
6
7  int main()
8  {  Time t;
9
10    t.hour = 7;
11    cout << "minuto = " << t.minute << endl;
12    return 0;
13 }
```

Parte III. Modifique o objeto **Time** para incluir o método **Tick** que incrementa o horário armazenado no objeto **Time** em um segundo. O objeto deve permanecer sempre em um estado consistente. Não esqueça de definir pré- e pós-condições. Utilize o programa seguinte para testar o novo método **Tick** em um *loop* que imprime no formato militar o horário em cada iteração, mostrando que o novo método funciona corretamente. Tenha certeza de testar os seguintes casos (altere os valores iniciais na linha 11):

f. incrementar para o próximo minuto;
g. incrementar para a próxima hora;
h. incrementar para o próximo dia.

```
1  // dtime3.cpp
2  // Driver para testar metodo Tick do tipo de dados Time
3  #include <iostream>
4  #include "time1.h"
5  using namespace std;
6
7  int main()
8  {  int i;
9     Time dinnerTime;
10
11    dinnerTime.setTime(0,0,0);
12    cout << "O horario inicial eh ";
13    dinnerTime.printMilitary();
14    cout << " horario militar" << endl;
15
16    for(i=1; i<=100; i++)
17    { dinnerTime.Tick();
18      dinnerTime.printMilitary();
19      cout << endl;
20    }
21    return 0;
22 }
```