	Universidade Federal de Campina Grande **Departamento de Sistemas e Computação** **Disciplina: *Técnicas de Programação*** **PROVA DO MÓDULO I – PRÁTICA – TURMA: 02** **SEMESTRE 2014.2 – DATA: 02/11/2014**	
Matrícula	**Nome**	**Nota**

1. Implementar uma estrutura que armazene inteiros, dos quais os dois primeiros correspondam a dois números quaisquer, enquanto os demais correspondam a resultados de seguintes operações *bit* a *bit*: (i) *NÃO* (*NOT*); (ii) *NÃO E* (*NAND*); (iii) *NÃO OU* (*NOR*); (iv) *NÃO OU EXCLUSIVO* (*NOT EXCLUSIVE OR*); (v) *DESLOCAMENTO DE N PARA A ESQUERDA* (*N-LEFT SHIFT*); e (vi) *DESLOCAMENTO DE N PARA A DIREITA* (*N-RIGHT SHIFT*). Os resultados de tais operações deverão ser inicializados por funções que recebam a referência à estrutura e inicializem seus membros. **1,2**

   Detalhes de implementação:

   - Deverão ser criadas as funções *not(struct bitwise *est)*, *nand(struct bitwise *est)*, *nor(struct bitwise *est)*, *notxor(struct bitwise *est)*, *n_lefts(struct bitwise *est)* e *n_rights(struct bitwise *est)*;
   - As operações bit-a-bit deverão ser testadas a partir de um programa que solicite do usuário os operandos e a operação desejada;
   - O usuário será capaz de solicitar tantas operações com pares de operandos quantas queira e de encerrar o programa de teste quando lhe for conveniente; e
   - Todos os resultados das operações deverão ser gravados em um arquivo.

2. O trecho de código a seguir representa o cabeçalho (*racional.h*) que descreve uma classe destinada a manipular números racionais.

```
#ifndef RACIONAL_H_
#define RACIONAL_H_
class Racional
{
    int n, d;
public:
    Racional(int num, int den);
    Racional(int num); // altera den para 1
    Racional(); // altera num para 0 e den para 1
    friend Racional operator+(const Racional&, const Racional&);
    friend Racional operator-(const Racional&, const Racional&);
    friend Racional operator*(const Racional&, const Racional&);
    friend Racional operator/(const Racional&, const Racional&);
    friend bool operator<(const Racional&, const Racional&);
    friend bool operator<=(const Racional&, const Racional&);
    friend bool operator>(const Racional&, const Racional&);
    friend bool operator >=(const Racional&, const Racional&);
    friend bool operator ==(const Racional&, const Racional&);
    friend ostream& operator <<(ostream&, const Racional&);
    friend istream& operator >>(istream&, Racional&);
};
#endif
```

Construir a implementação correspondente (*racional.cpp*) e testá-la com uma aplicação (*aplicação.cpp*). **1,4**

Detalhes de implementação:

- As operações deverão ser testadas a partir de um programa que solicite do usuário os operandos e a operação desejada;
- O usuário deverá ser capaz de solicitar tantas operações com pares de operandos quantas queira e de encerrar o programa de teste quando lhe for conveniente; e
- Todos os resultados das operações deverão ser gravados em um arquivo.

3. A fim de automatizar o processo de venda de sucos, uma cantina da UFCG solicitou da Turma 02 de Técnicas de Programação do semestre letivo 2014.2 a construção de um programa que implemente uma máquina de sucos na qual estão disponíveis os sabores laranja, limão, manga e cajá. O programa deverá:

   1. Exibir ao usuário os diferentes produtos vendidos pela máquina de sucos;
   2. Permitir ao usuário selecionar o produto desejado;
   3. Exibir ao usuário o preço do item selecionado;
   4. Receber dinheiro do usuário; e
   5. Entregar o item comprado ao usuário.

   Entrada A seleção do item desejado e seu preço.
   Saída O item selecionado.

   Considerar que a máquina terá dois componentes principais: Uma caixa registradora e vários distribuidores automáticos para conter e liberar os produtos. Os dois componentes deverão ser implementados por duas classes, definidas pelos trechos de código a seguir.

```
class CaixaRegistradora
{
   int  cashOnHand;                    // atributo que armazena o valor na caixa
                                       // registradora

 public:
   int  getTotalAtual()  const;        // Funcao para mostrar o total atual da caixa
                                       // registradora
                                       // Pos-condicao: Retorno do valor de TotalCaixa
   void  aceitaValor(int  EntraValor); // Funcao para  receber o valor  depositado
                                       // pelo usuário e atualizar o total da caixa
                                       // registradora
                                       // Pos-condicao: O total atualizado
   CaixaRegistradora(int  RecebeValor =  10);
                                       // Construtor que altera o valor do registrador
                                       // Pos-condicao: O total do caixa incrementado
                                       // do valor recebido (o valor default deverah
                                       //  ser  assumido  se  nenhum  valor  for
                                       especificado
};
```

```
class  Distribuidor
{
   int  NumItems;=                        // atributo que armazena o numero de items  no
                                          // distribuidor
   int  cost;                             // atributo que armazena o preco de um item

 public:
   int  getNumItems()  const;             // Funcao que exibe o numero de itens na maquina
                                          // Pos-condicao: Retorno do valor de NumItems
   int  getPreco()  const;                // Funcao que exibe o preço do item
                                          // Pos-condicao: Retorno do valor do preco
   void  fazVenda();                      // Funcao que decrementa o numero de itens
   Distribuidor (int  setNumItems  =  50,  int  setPreco  =  50);
                                          // Construtor que ajusta o preco e o numero
                                          // de itens no distribuidor para os valores
                                          //especificados pelo usuario
                                          // Pos-condicao: o preco e o numero de itens
                                          // ajustados
};
```

Escrever um programa em C++ que implemente e teste esta máquina de sucos. **1,4**

ÊXITO!