UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO E AUTOMAÇÃO

PROGRAMAÇÃO AVANÇADA – 1ª unidade/1ª avaliação (classes)
PROFESSOR: ADELARDO ADELINO DANTAS DE MEDEIROS

ALUNO: _____ MATRÍCULA: _____

Nesta avaliação, você deve desenvolver, em linguagem C++, um programa para representar e manipular linhas em um plano bidimensional. Uma linha é uma sequência ordenada de pontos, sendo cada ponto definido por suas coordenadas x e y. O programa deve ser construído da seguinte forma:

1. Defina um tipo de dados (uma `class`) adequado para representar um ponto bidimensional, denominado `Ponto`. As coordenadas do ponto podem assumir qualquer valor real.

2. Preveja ao menos as seguintes funcionalidades para a classe `Ponto`:
   - Sobrecarregue o `operator>>` para ler as coordenadas de um `Ponto` a partir de valores fornecidos via teclado.
   - Sobrecarregue o `operator<<` para imprimir um `Ponto` em tela, no formato [x;y], tal como no exemplo a seguir:
     `[-1.5;2]`

3. Utilizando o tipo `Ponto`, defina um tipo de dados (uma `class`) adequado para representar uma linha bidimensional, denominado `Linha`. Uma linha é composta por uma sequência de pontos, armazenada em um array de `Ponto`'s.

4. Preveja ao menos as seguintes funcionalidades para a classe `Linha`:
   - Sobrecarregue o `operator<<` para imprimir uma `Linha` em tela, usando o `operator<<` da classe `Ponto`. Deve ser impressa toda a sequência de pontos, tal como no exemplo a seguir para uma linha de 3 pontos:
     `[0;0][-1.5;2][7;-3.7]`
   - Preveja o método `adicionar` para acrescentar um `Ponto` a uma `Linha`. O novo ponto deve entrar no fim da linha, após os pontos já existentes. Lembre-se que, para adicionar um novo ponto, é necessário:
     a. Alocar nova área de memória, com capacidade para mais um ponto.
     b. Copiar os pontos antigos para a nova área.
     c. Liberar a área de memória antiga.
     d. Fazer com que a `Linha` aponte para a nova área de memória.
   - Sobrecarregue o `operator+` para concatenar duas linhas, originando uma linha definida pela união dos pontos de ambas. Por exemplo, se a primeira linha tem 3 pontos:
     `[1;2][0;0][-3;0.5]`
     e a segunda linha, 2 pontos:
     `[2;2][0;-2]`
     então a linha concatenada (primeira + segunda) deverá ter 5 pontos:
     `[1;2][0;0][-3;0.5][2;2][0;-2]`

5. Finalmente, faça um programa principal que utilize as funções programadas anteriormente para:
   - Criar 3 linhas: L1, L2 e L3.
   - Definir via teclado o número de pontos da linha L1.
   - Ler via teclado os valores das coordenadas dos pontos de L1.
   - Fazer L2 inicialmente igual a L1.
   - Definir via teclado o número de pontos adicionais a serem acrescentados na linha L2.
   - Ler via teclado os valores das coordenadas dos pontos adicionais de L2.
   - Fazer L3 igual à concatenação ("soma") de L1 e L2 (L3=L1+L2);
   - Imprimir as linhas L1, L2 e L3.

Apresenta-se a seguir um esboço do código em C++ que pode ser utilizado para iniciar o desenvolvimento. O programa principal (`main`) deverá ser exatamente igual ao apresentado, sem nenhuma modificação, acréscimo ou supressão.

```
#include <iostream>

using namespace std;

class Ponto
{
  ...
};

class Linha
{
  ...
};

...

int main(void)
{
  Linha L1,L2,L3;
  Ponto P;
  unsigned NPt, NPtAdic, i;

  do
  {
    cout << "Numero de pontos da 1a linha (>0): ";
    cin >> NPt;
  } while (NPt<=0);

  for (i=0; i<NPt; i++)
  {
    cout << "Digite o ponto indice " << i << ": ";
    cin >> P;
    L1.adicionar(P);
  }

  L2 = L1;

  do
  {
    cout << "Numero de pontos adicionais da 2a linha (>0): ";
    cin >> NPtAdic;
  } while (NPtAdic<=0);

  for (i=0; i<NPtAdic; i++)
  {
    cout << "Digite o ponto indice " << i+NPt << ": ";
    cin >> P;
    L2.adicionar(P);
  }

  L3 = L1+L2;

  cout << "1a linha: " << L1 << endl;
  cout << "2a linha: " << L2 << endl;
  cout << "3a linha: " << L3 << endl;

  return 0;
}
```