```python
#/usr/bin/python
# _*_ coding: utf-8

#
# Implementación del tipo de dato "lista de notas"
#

# Calcula el promedio de una lista de notas
# Entrada: la lista de notas
# Salida: el promedio de las notas redondeado a un decimal
def calculaPromedio(listaNotas):
   return round(sum(listaNotas)/len(listaNotas), 1)

# Determina si todas las notas en una lista de notas son suficientes
# Entrada: la lista de notas
# Salida: True si todas las notas en la lista son mayores o iguales a
4.0,
#         o la lista de notas está vacía; False en caso contrario
def todasNotasSuficientes(listaNotas):
   for nota in listaNotas:
      if nota < 4.0:
         return False
   return True

# Permite eliminar la peor nota en una lista de notas; si la nota más
baja
# está repetida, se elimina la primera de ellas
# Entrada: la lista de notas
# Salida: una nueva lista de notas, con las notas de la lista original
con
#         la excepción de la nota más baja; intentar reducir una lista de
#         notas vacía produce un error
def eliminaPeorNota(listaNotas):
   assert len(listaNotas) > 0, "No se puede eliminar la peor nota de una
lista vacía"
   mejoresNotas = []
   peor = listaNotas[0]
   for i in range(1, len(listaNotas)):
      if listaNotas[i] < peor:
         mejoresNotas.append(peor)
         peor = listaNotas[i]
      else:
         mejoresNotas.append(listaNotas[i])
   return mejoresNotas
```

```
# Calcula el promedio semestral para una lista de notas
# Entrada: una lista con 5 notas
# Salida: promedio semestral según la lista; si las primeras 3 notas
parciales
#         son suficientes, su promedio es el promedio semestral; sino, si
#         el promedio de las 3 primeras notas parciales es mayor o igual
a
#         5.0, este promedio es el promedio semestral; sino, el promedio
#         semestral se calcula con las 4 mejores notas en la lista
def calculaPromedioSemestral(listaNotas):
   assert len(listaNotas) == 5, "Se necesitan 5 notas para calcular
promedio semestral"
   notasParciales = listaNotas[0:3]
   promedio = calculaPromedio(notasParciales)
   if promedio < 5.0:
      if not todasNotasSuficientes(notasParciales):
         promedio = calculaPromedio(eliminaPeorNota(listaNotas))
   return promedio


#
# Implementación del tipo de dato "nota parcial curso"
#


#
# Lee una nota parcial de un curso
# Entrada: nombre del archivo con la nota parcial del curso
# Salida: un diccionario con el RUN del estudiantes como llave y
#         nota parcial del estudiante como valor
# Requiere: cada linea del archivo de entrada contiene
#           el par <RUN estudiante> <Nota del estudiante> (separados por
#           un espacio)
def leeNotaParcialCurso(nombreArchivoEntrada):
   notaParcialCurso = dict()
   archivoEntrada = open(nombreArchivoEntrada, 'r')
   for lineaArchivoEntrada in archivoEntrada:
      listaPalabras = lineaArchivoEntrada.strip().split()
      runEstudiante = listaPalabras[0]
      notaEstudiante = float(listaPalabras[1])
      notaParcialCurso[runEstudiante] = notaEstudiante
   return notaParcialCurso
```

```
#
# Implementación del tipo de dato "notas curso"
#


#
# Crea una estructura notas curso
# Entrada: la nota parcial del curso con que se inicializa las notas del
curso
# Salida: un diccionario con el RUN del estudiantes como llave y
#         la lista de notas parciales del estudiante como valor; esta
lista
#         contiene la nota parcial entregada para cada estudiante
def creaNotasCurso(notaParcialCurso):
   notasCurso = dict()
   runs = notaParcialCurso.keys()
   for run in runs:
      notasEstudiante = []
      notasEstudiante.append(notaParcialCurso[run])
      notasCurso[run] = notasEstudiante
   return notasCurso

#
# Agrega una nota parcial a las notas de un curso
# Entrada: la nota parcial del curso a ser agregada
# Asegura: cada estudiante tiene otra nota parcial; si no había registro
#          anterior del estudiante, la nota faltante se llena con un 1.0
def agregaNotaParcial(notasCurso, notaParcialCurso):
   runAntiguos = set(notasCurso.keys())
   runNuevos = set(notaParcialCurso.keys())
   runs = runAntiguos.union(runNuevos)
   for run in runs:
      notasEstudiante = [1.0]
      if notasCurso.has_key(run):
         notasEstudiante = notasCurso[run]
      if notaParcialCurso.has_key(run):
         notasEstudiante.append(notaParcialCurso[run])
      else:
         notasEstudiante.append(1.0)
      notasCurso[run] = notasEstudiante
```

```
#
# Escribe las notas de un curso en un archivo de salida
# Entrada: el nombre del archivo de salida y una estructura notas curso
# Requiere: las notas del curso deben registrar 5 notas por cada
estudiante
# Asegura: se ha escrito un archivo en que cada línea contiene el RUN del
#          estudiante, sus notas parciales y el promedio semestral
#          (todos separados con un espacio); los RUNs en el archivo están
#          ordenados de menor a mayor
def escribeNotasCurso(nombreArchivoSalida, notasCurso):
   archivoSalida = open(nombreArchivoSalida, "w")
   runs = sorted(notasCurso.keys())
   for run in runs:
      # Escribe el RUN del estudiante
      archivoSalida.write(run)
      notasEstudiante = notasCurso[run]
      # Escribe las notas parciales del estudiante
      for nota in notasEstudiante:
         archivoSalida.write(" ")
         archivoSalida.write(str(nota))
      # Escribe el promedio semestral del estudiante
      promedio = calculaPromedioSemestral(notasEstudiante)
      archivoSalida.write(" ")
      archivoSalida.write(str(promedio))
      # Finaliza la línea en el archivo
      archivoSalida.write("\n")
   archivoSalida.close()



#
# Bloque principal
#

# Entrada de datos: obtiene notas parciales de los estudiantes
pep1 = leeNotaParcialCurso("pep1.txt")
pep2 = leeNotaParcialCurso("pep2.txt")
controles = leeNotaParcialCurso("controles.txt")
pa = leeNotaParcialCurso("pa.txt")

# Procesamiento: junta las notas del curso
notasSemestre = creaNotasCurso(pep1)
agregaNotaParcial(notasSemestre, pep2)
agregaNotaParcial(notasSemestre, controles)
agregaNotaParcial(notasSemestre, pa)
agregaNotaParcial(notasSemestre, pa)

# Salida de datos: escribe archivo con notas parciales y promedio
semestral
# de cada estudiante
escribeNotasCurso("finales.txt", notasSemestre)
```