Introdução:

Um **arquivo binário** pode armazenar dados de qualquer maneira possível (por ser binário, há a possibilidade de organizar os *bytes* de qualquer combinação possível). Contudo, em geral, é estabelecido uma ordem pela qual os *bytes* são armazenados, de maneira a manter certa organização no arquivo, e de forma que os programas possam ler e armazenar esses bytes de uma forma padronizada. Imagine, por exemplo, que se não fosse feita essa padronização, não seria possível escrever programas que exibissem uma imagem no formato PNG na tela. Se não fosse padronizado o formato PNG, cada programa ia escrever e ler o arquivo binário de uma maneira diferente, ou seja, uma grande bagunça. Mas o PNG é um formato de arquivo binário, e foi PADRONIZADO que os *bytes* são organizados de uma maneira específica. Dessa forma, é possível que qualquer visualizador de imagens leia arquivos PNG e interpretem os bytes ali armazenados. Aqui na disciplina também vamos trabalhar com arquivos binários: não o PNG, mas arquivos binários que armazenam registros de dados.

Descrição:

Você deve implementar um programa que recebe como entrada o nome de um arquivo no formato binário. Seu programa deve abrir o arquivo para leitura, realizar a leitura desse arquivo e retornar todos os registros cujo campo “IDADE” seja maior ou igual a 18. Os registros do arquivo binário em questão são organizados da seguinte maneira:

ID-DO-USUARIO (4 bytes, inteiro), NOME (50 bytes, char), IDADE (4 bytes, inteiro)

Observe que os campos “ID-DO-USUARIO” e “IDADE” são inteiros (4 bytes). Então eles devem ser lidos para uma variável do tipo “int” do C.

Observe que o campo “NOME” é uma *string do C* que pode variar de tamanho, mas tem sempre no máximo 50 caracteres. Sendo assim, é um campo fixo de 50 bytes. Mas esse campo tem também o terminador de *string* do C (o ‘\0’), que vai dizer para você se o nome ali dentro ocupa 0 bytes (nulo), 1 byte, 30 bytes ou 49 bytes, por exemplo. Não é necessário tratar nada na leitura nesse caso então: aloca na memória RAM uma string de tamanho 50 chars, e lê para essa string toda do disco de tamanho 50 bytes. Observe que como no disco o ‘\0’ já tá armazenado, quando você realizar a leitura para a memória RAM, sua string também terá o ‘\0’, então não é necessário tratar isso. Quando o campo “NOME” for nulo, ele começa imediatamente pelo ‘\0’, ou seja, o strlen ou tamanho da string é zero.

Entrada:

O nome de um arquivo (que vai estar no formato binário padronizado acima).

Saída:

Todos os registros lidos desse arquivo cujo campo “IDADE” seja maior ou igual a 18.

A resposta do seu programa para cada registro deve estar no formato abaixo se o campo “NOME” não for nulo:

O usuario NOME-DO-USUARIO eh de maior.\n

Ou no formato abaixo se o campo “NOME” for nulo:

O usuario de identificacao ID-DO-USUARIO eh de maior.\n

Exemplo:

Considere o arquivo de nome **registros.bin** que contém os registros abaixo armazenados de forma binária (não é um arquivo de texto, isso é só uma representação visual desse arquivo!):

id 1 – Joao – 30 anos
id 2 – (nulo) – 15 anos
id 3 – Maria – 28 anos
id 4 – (nulo) – 24 anos
id 5 – Jose – 56 anos
id 6 – Leticia – 10 anos

A **entrada** do seu programa será:

registros.bin

A **saída** do seu programa deve ser (considere '\n' uma representação visual do pulo de linha):

O usuario Joao eh de maior.\n
O usuario Maria eh de maior.\n
O usuario de identificacao 4 eh de maior.\n
O usuario Jose eh de maior.\n