Sistemas Operacionais:
Exercício 1:

- exemplo.c:
  Nesse exercício o aluno deverá implementar 2 formas de criar uma variável, tanto de forma dinâmica usando malloc quanto local, e pedir para retornar o endereço dessa variável para a função main. Em seguida tentar acessá-la em ambos os casos. Perguntar por que em um caso não ocorre erro enquanto em outro dá segmentation fault.

Exercício 2:

- main_fatorial_recursivo.c:
  Pedir para que o aluno implemente o fatorial na sua versão recursiva. Fornecer uma função que faz o print da pilha de chamadas (ou usar o python tutor) e perguntar quantas chamadas foram feitas, perguntar por que não é possível acessar uma variável inicializada na chamada anterior da função fatorial a menos que passada a referência, possível problema com essa implementação. Perguntar também o significado do offset apresentado na pilha.

Exercício 3:

- main_fatorial_nao_recursivo.c, fatorial_aux.c, fatorial.h:
  Aqui o aluno é perguntado para implementar a versão não recursiva do algoritmo, mas usando arquivos separados da main. Novamente, pede-se para que o aluno visualize a pilha de execução para responder quantas vezes foi chamada a função e citar as vantagens em relação a versão recursiva.

Exercício 4:

- main_strcpy.c, strcpy.c, strcpy.h:
  Aqui é pedido que o aluno implemente um protótipo da função strcpy, mas restringindo o uso da biblioteca string.h. A ideia é fixar que toda string em C deve terminar com '\0', a noção de aritmética de ponteiros e a passagem de parâmetros via argv por linha de comando.

Exercício 5:

- main_strlib.c, strlib.c, strlib.h:
  Aqui o aluno deveria implementar 3 funções básicas em uma biblioteca separada:
  1. Strlen
  2. Strcpy
  3. Split

  Em seguida, o aluno deveria criar um array com ponteiros para essas funções e requisitar do usuário que digite uma palavra e uma opção dentre as 3 funções implementadas, sendo acessadas pelo ponteiro de função. O objetivo era preparar o aluno com entradas do usuário via scanf, apresentar a função atoi e preparar para quando viesse uma questão posterior a respeito de um processo recebendo um sinal e precisando tratá-lo, acessando um vetor de funções. Como as funções mencionadas acima possuem assinaturas distintas a respeito da quantidade de parâmetros e tipo, uma solução é um wrapper, com a biblioteca stdarg.h. Um exemplo de wrapper poderia ser passado aos alunos, dado que esse não é o objetivo principal desse exercício, ou então pedir para ao

invés de criar uma biblioteca de string, criar uma que faz manipulações entre dois números (soma, subtração etc).

- Exercício 6:
  Pedir para que o aluno implemente a alocação de memória para 1.000.000 posições, uma como uma matriz de 10000x100 e outra como um único vetor. Pedir para o aluno inicializar todas as posições com o valor 1 em cada um dos casos e perguntar em qual foi mais lento e porquê. Perguntar se o endereço da posição m[i][0] é consecutivo ao m[i-1][99]. Aqui o aluno se habitua com o acesso a vetores dinâmicos e o conceito de ponteiro para ponteiro. Uma possibilidade não explorada é colocar uma situação na qual deve ser usado a função realloc, pedir para explicar o que acontece com o endereço inicial do vetor quando a quantidade de memória extra necessária é pequena e quando é grande.