O código fonte (arquivo ".c" funcional, que possa ser executado para testar) deve ser enviado via Moodle conforme data e horário especificados na atividade. O trabalho representa 30% da primeira nota para quem fizer e apresentar. O código deve ser desenvolvido na linguagem de programação C, em grupos de até 3 (TRÊS, o que significa que são três no máximo!) pessoas. Casos com grupos com mais de três integrantes, e/ou soluções iguais entre dois grupos, serão considerados trabalhos entregues e receberão a nota 0 (zero).

O conteúdo entregue será apresentado, em horário a combinar. Na apresentação, poderá ser solicitado que cada membro do grupo apresente algum código ou trecho de código individualmente, conforme ESCOLHA DO PROFESSOR. A entrega dos exercícios será desconsiderada para aqueles alunos que não apresentarem o código do exercício definido pelo professor. Portanto, é importante que todos os membros do grupo tenham domínio sobre todo o conteúdo entregue.

Considere uma memória imaginária de 1024 bytes, endereçados de 0 até 1023. Um sistema criado para gerenciar a alocação desta memória trabalha com uma lista encadeada CIRCULAR em que cada elemento contém os seguintes dados:

- um bit para indicar se aquela parte da memória está em uso (1) ou não (0);

- o nome do processo que está armazenado ali (usem um char), no caso de áreas ocupadas;

- um inteiro para indicar o endereço onde começa a parte da memória representada por aquele elemento;

- um inteiro com o tamanho da parte representada pelo elemento;

- um ponteiro para o próximo elemento da lista.

Inicialmente, a lista contém apenas um elemento, com os valores

<0, -, 0, 1024, &>

indicando que no momento a memória pode ser considerada um bloco único, desocupado, que começa em 0 e tem tamanho 1024. O nome está representado por '-' pois área vazia não recebe nome. Ao ser criado um processo (A), uma parte da memória será alocada para ele, e a lista ficará, por exemplo,

<1, A, 0, 5, &>, <0, -, 5, 1019, &>

Neste caso, o símbolo & representa que agora ali está armazenado o endereço do próximo da lista. Eventualmente, após uma série de alocações e liberações, a lista pode ficar com diversos elementos, intercalando alguns que representam áreas alocadas e outros que representam áreas livres. Esta é uma das formas que podem ser usadas para manter um controle das áreas livres e alocadas da memória.

Façam um programa que gerencie esta lista. No caso do programa do trabalho, porém, não serão chamadas do sistema operacional, mas solicitações do usuário que causarão alterações nesta lista. O programa deve ter um menu que permita ao usuário escolher, a cada ciclo, que ação deseja executar dentre as opções abaixo descritas:

1) liberar área alocada: o usuário informa o nome de um processo e o bloco da área de memória que contém este processo é liberado. Isto implica na fusão da área do bloco com eventuais áreas livres adjacentes;

2) incluir processo com o método first-fit: o usuário informa o nome de um processo e o seu tamanho e o programa cria um bloco para ele na primeira área livre com tamanho suficiente para contê-lo;

3) incluir processo com o método best-fit: o usuário informa o nome de um processo e o seu tamanho e o programa cria um bloco para ele na área livre com tamanho mais próximo do tamanho necessário;

4) incluir processo com o método worst-fit: o usuário informa o nome de um processo e o seu tamanho e o programa cria um bloco para ele na área livre com tamanho mais distante do mínimo necessário.

5) Informar ao usuário que há problema de fragmentação – informar que não foi possível alocar por não haver um bloco com tamanho suficiente para conter o processo, mas avisar que agrupando blocos daria para alocar.

6) fazer a desfragmentação, agrupando os blocos livres em um apenas, no final da lista.

7) mostrar a situação da lista, um bloco por linha.

Em todas as alocações, se a área livre não for do tamanho exato, deve ser dividida em um bloco para o processo e um bloco com o resto da área livre.