ESTRATÉGIAS UTILIZADAS PARA IMPLEMENTAÇÃO

get_brk: Retorna o valor atual de brk, é uma função auxiliar apenas para aumentar a legibilidade

setup_brk: Inicia alocador, salvando o endereço atual de brk em duas variáveis, BRK_INICIAL e BRK_ATUAL. A variável BRK_INICIAL não é alterada, apenas a BRK_ATUAL é alterada a medida que novos blocos são alocados

dismiss_brk: Encerra alocador, retornando brk ao endereço inicial salvo em BRK_INICIAL

memory_alloc:Faz a alocação dos blocos utilizando método de first fit de acordo com o enunciado, recebe um tamanho e retorna o endereço do bloco de dados para serem utilizados. Também faz a administração de uso, e quebra de um bloco livre maior em dois blocos menores quando sobra espaço (um livre, e outro ocupado).

memory_free: Testa se o endereço do bloco de dados informado está entre o endereço de BRK_INICIAL e o BRK_ATUAL, caso esteja, o uso é setado para 0, para indicar que o bloco está livre. Após liberar o bloco, passa pela heap conectando os blocos livres consecutivos em blocos livres maiores.

Correção da fragmentação de memória: Na função memory_alloc, quando é encontrado um bloco maior que o bloco que estamos alocando, esse bloco é dividido em dois blocos menores, um utilizado e outro livre em seguida, se sobrar espaço o suficiente para as informações gerenciais. E na função free, após mudar o USO de um bloco para 0, passa pela heap conectando blocos vazios consecutivos em blocos maiores.
No arquivo main.c de teste, para demonstrar que a correção da fragmentação está funcionando basta descomentar as linhas de código no final, e checar o tamanho de cada bloco após cada free, no comportamento correto o primeiro bloco aumenta de tamanho, enquanto os outros permanecem aparentemente iguais nos prints, pois seus tamanhos estão salvos em uma região de memória, agora desalocada, ou seja estamos printando essencialmente um lixo de memória que está dentro do bloco livre.