Programação de Software Básico - Trabalho 2

1 Introdução

O segundo trabalho da disciplina consiste no desenvolvimento de uma biblioteca de gerenciamento de memória simples em C, que permite a alocação e liberação de memória em um *pool* pré-alocado. Esse *pool* será gerenciado por uma estrutura de dados associada, que organizará as regiões de alocação.

2 Especificação

2.1 Estruturas de Dados

allocation_t: Uma estrutura que representa um bloco de memória alocado. Deve conter os seguintes campos:

- start: Um ponteiro para o início do bloco alocado.
- size: O tamanho da alocação.
- next: Um ponteiro para a próxima allocation_t (para formar uma lista encadeada).

```
/* mymemory.h */
typedef struct allocation {
      void *start;
      size_t size;
      struct allocation *next;
} allocation_t;
```

mymemory_t: Uma estrutura que representa o *pool* total de memória e todas as alocações atuais. Deve conter os seguintes campos:

- pool: Um ponteiro para o início do bloco de memória total.
- total_size: O tamanho total do bloco de memória.

- head: Um ponteiro para a primeira allocation_t (cabeça da lista encadeada).

```
/* mymemory.h */
typedef struct {
      void *pool;     // ponteiro para o bloco de memória real
      size_t total_size;
      allocation_t *head; // ponteiro para lista encadeada
} mymemory_t;
```

2.2 Funções Principais

- *mymemory_t* mymemory_init(size_t size)*: Aloca o bloco de memória total e retorna um ponteiro para ele.
- *void* mymemory_alloc(mymemory_t *memory, size_t size)*: Tenta alocar um bloco de memória de tamanho especificado. Se a alocação for bem-sucedida, retorna um ponteiro para o início do bloco. Caso contrário, retorna NULL.
- *void mymemory_free(mymemory_t *memory, void *ptr)*: Libera a alocação apontada por *ptr*. Se *ptr* não for uma alocação válida, a função não deve fazer nada.
- *void mymemory_display(mymemory_t *memory)*: Exibe todas as alocações atuais, incluindo o início e o tamanho de cada alocação.
- *void mymemory_stats(mymemory_t *memory)*: Exibe estatísticas gerais sobre a memória, incluindo:
  - Número total de alocações
  - Memória total alocada (em bytes)
  - Memória total livre (em bytes)
  - O maior bloco contíguo de memória livre
  - Número de fragmentos de memória livre (ou seja, blocos de memória entre alocações)
- *void mymemory_cleanup(mymemory_t *memory)*: Libera todos os recursos (incluindo todas as alocações e o bloco de memória total).

2.3 Estratégias de Alocação

Você pode escolher diferentes estratégias:

- Primeiro Ajuste (*First Fit*): A primeira lacuna suficientemente grande.
- Melhor Ajuste (*Best Fit*): A menor lacuna suficientemente grande.
- Pior Ajuste (*Worst Fit*): A maior lacuna.

Para cada estratégia, você ajustará a função *mymemory_alloc()* para que ela procure os espaços disponíveis de acordo. Para verificação do funcionamento do alocador, deve-se utilizar um conjunto de blocos de tamanho variado (por exemplo, 10, 25, 50, 100, 250, 500, 1000 ...) e variar a ordem das alocações. Recomenda-se um conjunto de aproximadamente 50 blocos.

2.4 Restrições

Não é permitido usar funções de alocação global (como *malloc()* e *free()*) a não ser para a alocação do *pool* de memória e do gerenciamento das estruturas de dados de controle (*mymemory_t* e nodos da lista (*allocation_t*)). O seu alocador de memória deverá funcionar adequadamente, retornando endereços válidos para regiões alocadas.

3 Avaliação

Este trabalho deverá ser realizado em grupos de até três integrantes e apresentado em sala de aula na data indicada no Moodle (apresentação em torno de 10 minutos). Para a entrega, é esperado que apenas um dos integrantes envie pelo Moodle, até a data e hora especificadas, um arquivo *.tar.gz* ou *.zip* do projeto contendo o código fonte desenvolvido.

Critérios de Avaliação

- Funcionalidade: Seu código deve compilar sem erros ou avisos e todas as funções devem funcionar conforme especificado.
- Eficiência: O gerenciador de memória deve fazer uso eficiente do espaço, com pouca fragmentação, e não deve ter vazamentos de memória (*memory leaks*).
- Apresentação: Além do código, você deve ser capaz de explicar em detalhes o código fonte criado durante a apresentação.

Observações

- A cópia parcial ou completa do trabalho terá como conseqüência a atribuição de nota ZERO ao trabalho dos alunos envolvidos.
- A cópia de código ou algoritmos existentes da Internet também não é permitida. Se alguma ideia encontrada na rede for utilizada na implementação, a referência deve constar no código em um comentário.
- Caso seja utilizado ajuda de colegas ou de alguma ferramenta de inteligencia artificial (IA), explicar como foi utilizado. Trabalhos que foram gerados inteiramente por IA e que os integrantes não conseguirem explicar em detalhes seu funcionamento, receberão grau zero.