Informações gerais

Neste laboratório, você será encarregado de implementar um sistema de atendimento em uma clínica médica que precisa organizar as consultas de pacientes com base em diferentes critérios de prioridade. O objetivo é implementar um sistema eficiente usando uma fila de prioridades baseada em vetor e algoritmos de ordenação. Atenção aos seguintes requisitos:

1. **Prioridade por Gravidade**:
    - Cada paciente é classificado com uma gravidade, representada por um número inteiro. Quanto menor o número, maior a gravidade.
    - O sistema deve organizar as consultas de acordo com a gravidade, priorizando os pacientes com gravidade mais alta.
2. **Prioridade por Ordem de Chegada**:
    - Em casos de pacientes com a mesma gravidade, a ordem de chegada deve ser considerada para determinar a prioridade.
3. **Adição e Remoção de Pacientes**:
    - O sistema deve permitir a adição de novos pacientes à fila de espera.
    - Pacientes podem ser removidos da fila quando atendidos.
4. **Atualização de Gravidade**:
    - Em alguns casos, a gravidade do paciente pode mudar (por exemplo, um paciente que estava estável pode piorar).
    - O sistema deve permitir a atualização da gravidade do paciente e reorganizar a fila conforme necessário.
5. **Relatório de Atendimento**:
    - O sistema deve ser capaz de gerar relatórios indicando a ordem de atendimento, com informações sobre o nome do paciente, gravidade e horário de chegada.

Observações importantes:

1. Neste laboratório será permitido o uso apenas das bibliotecas `stdio.h`, `stdlib.h`, `stdlib.h` e `string.h`.
2. Para compilar sua solução, utilize o `Makefile`, chamando no terminal o comando `make`.

TAD para manipulação do sistema de atendimento

Neste projeto, você irá desenvolver e gerenciar um sistema de atendimento para uma clínica médica, utilizando o conceito de Fila de Prioridade e técnicas de Ordenação para garantir que os pacientes sejam atendidos de acordo com a gravidade de seus casos e o horário de chegada. A representação de cada paciente na fila é feita através de uma estrutura contendo informações como o nome, a gravidade do caso e outra estrutura para representar o horário de chegada, com hora e minuto. Essa organização assegura que a lista esteja sempre ordenada, priorizando o atendimento da forma mais eficiente possível. A estrutura de dados para o Item é definida como segue:

- **Nome do paciente**
- **Gravidade**
- **Hora de chegada**

Com esta estrutura estabelecida, você implementará as seguintes operações fundamentais para o gerenciamento da fila de prioridade:

- **p_fp criar_filaprio(int tam)** - Cria a fila de prioridade com tamanho definido
- **void insere(p_fp fprio, Item item)** - Insere um Item na fila
- **Item extrai_maximo(p_fp fprio)** - Extrai o Item de maior prioridade

- **int vazia(p_fp fprio);** - Verifica se a fila está vazia
- **int cheia(p_fp fprio)** - Verifica se a fila está cheia
- **void atualizar_gravidade(p_fp fprio, char nome[], int nova_gravidade)** - Atualiza a gravidade de um paciente e reordena a fila
- **void destroi_fila(p_fp fprio)** - Libera a fila de memória

---

Questão 1

As operações e seus comandos de entrada correspondentes são:

- **C Tamanho**: Cria uma fila com a capacidade da clínica médica.
- **I Nome Gravidade Hora de Chegada**: Insere elemento.
- **R**: Atende o próximo paciente da fila e o remove, indica qual o paciente atendido.
- **A Nome Nova Gravidade**: Atualiza a gravidade de um paciente.
- **P**: Exibe a fila de prioridade atual.
- **F**: Fim do programa.

Exemplo de entrada padrão:

```
C 4
R
I João 3 10:00
I Maria 1 10:15
I Ana 2 10:30
I Pedro 3 10:45
I Gabriel 2 11:00
P
A Pedro 1
P
R
F
```

Saída esperada para o exemplo fornecido:

```
Fila vazia
Fila cheia
1. Paciente: Maria, Gravidade: 1, Chegada: 10:15
2. Paciente: Ana, Gravidade: 2, Chegada: 10:30
3. Paciente: João, Gravidade: 3, Chegada: 10:00
4. Paciente: Pedro, Gravidade: 3, Chegada: 10:45
```

```
1. Paciente: Maria, Gravidade: 1, Chegada: 10:15
2. Paciente: Pedro, Gravidade: 1, Chegada: 10:45
3. Paciente: Ana, Gravidade: 2, Chegada: 10:30
4. Paciente: João, Gravidade: 3, Chegada: 10:00

Atendimento: Maria
```

Pode-se notar que ocorreu o seguinte no exemplo: criou-se uma fila com capacidade para 4 pacientes. Na primeira interação, houve uma tentativa de atendimento a um paciente, mas a fila estava vazia. Posteriormente, foram inseridos 5 pacientes, entretanto, ao tentar adicionar o quinto, foi informado que a fila estava cheia. Realizou-se a impressão da lista atual, mantendo-a sempre ordenada. Em seguida, procedeu-se à atualização da gravidade do paciente Pedro e, posteriormente, imprimiu-se a nova configuração da lista. Por fim, foi realizado o atendimento da próxima pessoa na fila, no caso, Maria.

Após codificar o cliente e também as operações necessárias na implementação, compile tudo utilizando o `Makefile` e teste executando o seguinte comando:

```
./cliente.bin < teste_Q1.in
```

Onde `teste_Q1.in` é um arquivo contendo uma sequência de comandos como a exemplificada acima.