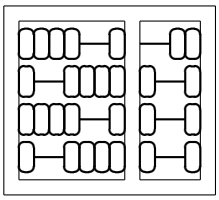

MC202 – Laboratório 12
Filas de prioridade utilizando Heaps
Instituto de Computação — UNICAMP
Prof.: Anderson Rocha
anderson.rocha@ic.unicamp.br

Objetivos

Entender e exercitar o funcionamento do *Heap* a partir da implementação de uma fila com prioridades.

Especificação

1. O programa deverá interpretar três tipos de operações diferentes: **criação** (C), **inserção** (I) e **remoção** (R). Tais operações serão indicadas no arquivo de entrada, representadas por uma letra e um número (numa linha) indicando a operação e a quantidade de registros para serem utilizados na operação, seguida dos registros (nas linhas seguintes).

2. Uma fila de prioridade é uma estrutura de dados que considera uma prioridade para cada elemento para armazená-lo e removê-lo (dar atendimento num banco, por exemplo). Ela pode ser implementada como um *Heap*, onde a prioridade de um elemento é dada como uma função de seus dados.

3. **A ordem de chegada deve ser levada em conta para pessoas com a mesma prioridade**.

4. O vetor para o *heap* deverá conter 500 posições.

5. A operação de remoção sempre dá preferência a elementos com maior prioridade. Para elementos de mesma prioridade deverá ser removido o primeiro elemento inserido na fila. Observe que não é possível entrar com o código do elemento a ser removido. Portanto, a operação R não tem parâmetros. Depois de ser removido, o elemento deverá ser impresso.

6. O tipo de dados que vai ser armazenado em cada nó da lista será um ponteiro do tipo *struct*, contendo os campos: `CPF`, `Nome`, `Sexo` e `Prioridade`, os quais deverão seguir o padrão CSV introduzido no Lab-03 bem como os limites de armazenamento mostrados a seguir:

```
struct pessoa {
   char cpf[12];
   char nome[100];
   char sexo;
   int prioridade;
};
```

Exemplo

Dada a entrada:

```
5
C 3
54637328498,Joao Paulo Miranda,M,3,
75487654329,Claudia Saboia Da Silva,F,1,
64548495746,Jose Gonsales,M,3,
I 4
75648484748,Carlos De Souza,M,4,
54746484848,Maria De Rezende,F,3,
53453453453,Jaime Dutra,M,5,
65754645645,Elisa Machado,F,3,
R
R
```

A saída esperada é:

```
53453453453,Jaime Dutra,M,

75648484748,Carlos De Souza,M,
```

Observações

1. Notem que esta atividade é praticamente idêntica ao Laboratório 6. As entradas e saídas serão as mesmas, porém a implementação deverá ser feita utilizando *Heap*. Quem submeter a atividade 3 como sendo a atividade 12 sofrerá sanções na disciplina.

2. Não será necessário escrever na tela instruções para o usuário digitar as entradas. Utilizar o redirecionamento de entrada e saída padrão para testar os executáveis.

   ```
   meu_programa.exe < arquivo_entrada.txt > teste_saida.txt
   ```

   ou seja, o que você digitaria no teclado deverá estar no primeiro arquivo e o que apareceria na tela vai estar no segundo arquivo.

3. Observe que apenas as operações C,I vão receber registros nas linhas seguintes do arquivo, sendo que a operação R não recebe registro.

4. As operações C,I não devem gerar nada no arquivo de saída.

5. Não é necessário validar os possíveis erros no arquivo de entrada: arquivo vazio, dados faltantes, dados repetidos, etc. Pode-se assumir que os dados vão estar corretos (inclusive nos testes fechados). No caso de dados repetidos (CPFs iguais, nomes iguais), simplesmente deverão ser inseridos no *heap* referente à pessoa com aquela mesma prioridade.

6. Deverão ser usadas funções para separar as funcionalidades do código.