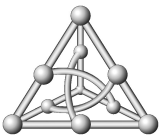

UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL
Faculdade de Computação

Estacionamento da FACOM

Trabalho 2

Algoritmos e Programação II

1 Descrição

A área de Ciência da Computação na UFMS teve seu ponto de partida em 1987, com a implantação do Curso de Bacharelado em Ciência da Computação, no Departamento de Matemática. Os pioneiros dessa implantação foram os Professores Edson Norberto Cáceres e Sergio Roberto de Freitas. Com o crescimento do grupo, o Departamento de Computação e Estatística (DCT) foi criado em 1992, deixando de existir com a implantação da Faculdade de Computação (FACOM) em 2009.

Desde sua criação, o DCT (hoje FACOM) contribuiu para a formação de bons alunos e pesquisadores. Essa boa formação pode ser atestada pelos inúmeros prêmios recebidos por alunos, ex-alunos ou professores de seu quadro. Além disso, a FACOM tem crescido muito nos últimos anos, alcançando mais de 50 professores e 1000 alunos, tendo hoje um edifício próprio e um estacionamento.

Pela quantidade de docentes, o gerenciamento desse estacionamento ao longo do tempo tem se tornado mais difícil, e por isso a FACOM precisa de nossa ajuda. As capivaras mais habilidosas da UFMS serão designadas para coordenar o estacionamento da FACOM. Contudo, as capivaras serão matriculadas em Algoritmos e Programação I apenas no próximo semestre, então você deve desenvolver um programa para efetuar o controle do estacionamento, que será reformulado, passando a utilizar o sistema de estacionamento em pilha (veja a Figura 1), e poderá ser visto de forma simplificada da seguinte maneira:

- Serão instaladas 3 pilhas (P0, P1 e P2), com suporte a no máximo 5 veículos cada;
- Ao chegar no estacionamento, um veículo é designado à pilha com menos carros no momento. Caso haja empate, escolha a pilha de menor número (P0 < P1 < P2);
- Um veículo não terá acesso ao estacionamento caso todas pilhas estejam cheias;
- Caso um veículo a ser retirado não esteja no topo[1] da pilha, remova os que estiverem abaixo, coloque-os em uma fila temporária, remova o veículo desejado, e então coloque os veículos da fila de volta na pilha[2].

[1] Nesse caso, o topo é a posição mais próxima ao solo
[2] Possivelmente a ordem dos veículos na pilha será alterada

2 Entrada e saída

A **entrada** contém vários casos de teste, cada um representando os registros de um dia de movimentação no estacionamento. A primeira linha contém um número inteiro $c > 0$ que se refere ao número de dias (casos de teste). Após esse número, uma linha em branco é apresentada, e a seguir, são apresentadas as informações dos c dias, em sequência.

Cada dia tem como entrada a data em específico, no formato DD/MM/AAAA. Depois, as operações do dia são apresentadas em ordem cronológica, uma em cada linha. Considere que uma PLACA tem o formato LLL-NNNN (onde L é uma letra maiúscula e N é um número de um dígito) e que as possíveis operações são:

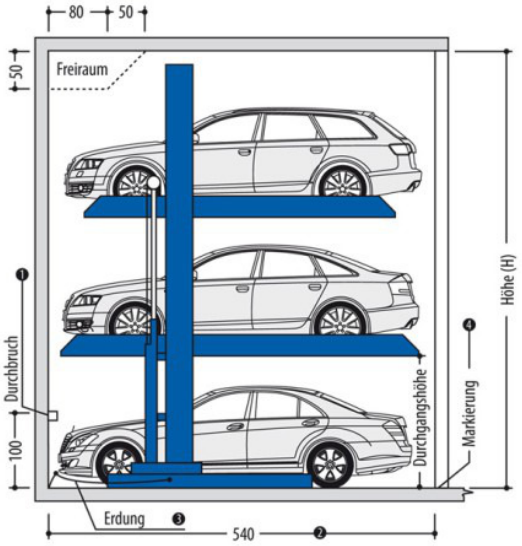

Figura 1: Uma pilha hidráulica de estacionamento

Operação	Descrição
`E PLACA`	Entrada: Um veículo com a placa indicada chegou
`S PLACA`	Saída: Um veículo com a placa indicada deve ser retirado
`I P?`	Imprima a lista dos carros na pilha P? (? = 0, 1 ou 2)
`F`	Operações do dia encerradas (limpe as pilhas)

Considere que cada dia contém apenas uma operação `FIM` e que esta é sempre a última operação do dia. Após o conjunto de operações de um dia, uma linha em branco é apresentada.

Na **saída**, para cada um dos casos de teste seu programa deve imprimir em uma linha a data do dia em questão. Nas linhas seguintes são apresentadas saídas referentes a cada uma das operações realizadas, uma em cada linha, de acordo com a tabela a seguir:

Operação	Saída impressa após a execução da operação
`E PLACA`	`C PLACA` caso o estacionamento esteja cheio `E PLACA` caso o veículo seja estacionado
`S PLACA`	`N PLACA` caso o veículo com a placa não exista `S PLACA` caso o veículo seja removido
`I P?`	Placas dos veículos da pilha P? do mais próximo ao solo para o mais alto, separadas por vírgula (ex: `P2:PLACA,PLACA,PLACA`)
`F`	`F` seguido de uma linha em branco

3 Exemplo de entrada

```
2

01/12/2013
E AAA-0000
E BBB-1111
E CCC-2222
E DDD-3333
S BBB-1111
I P0
I P1
I P2
F

02/12/2013
E HTG-0011
E HSO-1111
E HUV-2022
E HST-3133
I P0
I P1
I P2
E HSX-5555
E HBA-4545
E HUN-6789
I P0
S HTG-0011
I P0
S HTG-0011
F
```

4 Exemplo de saída

```
01/12/2013
E AAA-0000
E BBB-1111
E CCC-2222
E DDD-3333
S BBB-1111
P0:DDD-3333,AAA-0000
P1:
P2:CCC-2222
F

02/12/2013
E HTG-0011
E HSO-1111
E HUV-2022
E HST-3133
P0:HST-3133,HTG-0011
P1:HSO-1111
P2:HUV-2022
E HSX-5555
E HBA-4545
E HUN-6789
P0:HUN-6789,HST-3133,HTG-0011
S HTG-0011
P0:HST-3133,HUN-6789
N HTG-0011
F
```

5 Exigências

Você DEVE usar as definições a seguir em seu trabalho:

```
#define MAX_VEICULOS 5
#define NUM_PILHAS   3

/* Armazena informacoes de um veiculo estacionado em uma pilha */
typedef struct cel {
   char placa[9];
   struct cel *prox;
} veiculo;

/* Armazena informacoes de uma pilha */
typedef struct {
   int veiculos;  /* Quantidade de veiculos estacionados */
   veiculo *topo; /* Topo da pilha */
} pilha;
```

Além disso, como você pôde perceber pelo enunciado do trabalho, você DEVE utilizar em alguns momentos uma **fila** auxiliar para manobrar alguns veículos. Além disso, você pode optar por utilizar pilhas e filas com alocação encadeada **com** ou **sem** cabeça. O uso da estrutura de dados abaixo é sugerido (**opcional**):

```
/* Armazena informacoes do estacionamento */
typedef struct {
   char data[11];
   pilha P[NUM_PILHAS]; /* Armazena as pilhas P0, P1, ..., NUM_PILHAS-1 */
} estacionamento;
```

Caso alguma exigência não seja cumprida, a nota é ZERO.

6 Entrega

Instruções para entrega do seu trabalho:

1. **Cabeçalho**

   Seu trabalho deve ter um cabeçalho com o seguinte formato:

```
/*************************************************
 *
 * Nome do(a) estudante
 * Trabalho 2
 * Professor(a): Nome do(a) professor(a)
 *
 */
```

2. **Compilador**

   Os(as) professores(as) usam o compilador da linguagem C da coleção de compiladores GNU `gcc`, com as opções de compilação `-Wall -std=c99 -pedantic` para corrigir os programas. Se você usar algum outro compilador para desenvolver seu programa, antes de entregá-lo verifique se o seu programa tem extensão `.c`, compila sem mensagens de alerta e executa corretamente.

3. **Forma de entrega**

   A entrega será realizada diretamente na página da disciplina no AVA/UFMS. Um fórum de discussão deste trabalho já se encontra aberto. Após abrir uma sessão digitando seu *login* e sua senha, vá até o tópico "Trabalhos", e escolha "T2 - Entrega". Você pode fazer o upload de vários rascunhos, mas **o envio definitivo deve ser feito até a data indicada no AVA**. Apenas com o envio definitivo seu trabalho será corrigido. Encerrado o prazo, não serão mais aceitos trabalhos.

4. **Atrasos**

   Trabalhos atrasados não serão aceitos. Não deixe para entregar seu trabalho na última hora. Para prevenir imprevistos como queda de energia, problemas com o sistema, falha de conexão com a internet, sugerimos que a entrega do trabalho seja feita pelo menos um dia antes do prazo determinado.

5. **Erros**

   Trabalhos com erros de compilação receberão nota ZERO. Faça todos os testes necessários para garantir que seu programa está livre de erros de compilação.

6. **O que entregar?**

   Você deve entregar um único arquivo contendo APENAS o seu programa fonte com o mesmo nome de seu login no passaporte UFMS, como por exemplo, `fulano.silva.c`. NÃO entregue qualquer outro arquivo, tal como o programa executável, já compilado.

7. **Verificação dos dados de entrada**

   Não se preocupe com a verificação dos dados de entrada do seu programa. Seu programa não precisa fazer consistência dos dados de entrada. Isto significa que se, por exemplo, o seu programa pede um número entre 1 e 10 e o usuário digita um número negativo, uma letra, um cifrão, etc, o seu programa pode fazer qualquer coisa, como travar o computador ou encerrar a sua execução abruptamente com respostas erradas.

8. **Arquivo com o programa fonte**

   Seu arquivo contendo o programa fonte na linguagem C deve estar bem organizado. Um programa na linguagem C tem de ser muito bem compreendido por uma pessoa. Verifique se seu programa tem a indentação adequada, se não tem linhas muito longas, se tem variáveis com nomes significativos, entre outros. Não esqueça que um programa bem descrito e bem organizado é a chave de seu sucesso. Não esqueça da documentação de seu programa e de suas funções.

   Dê o nome do seu usuário do Passaporte UFMS para seu programa e adicione a extensão `.c` a este arquivo. Por exemplo, `fulano.silva.c` é um nome válido.

9. **Conduta Ética**

   O trabalho deve ser feito INDIVIDUALMENTE. Cada estudante tem responsabilidade sobre cópias de seu trabalho, mesmo que parciais. Não faça o trabalho em grupo e não compartilhe seu programa ou trechos de seu programa. Você pode consultar seus colegas para esclarecer dúvidas e discutir idéias sobre o trabalho, ao vivo ou no fórum de discussão da disciplina, mas NÃO copie o programa!

   Trabalhos envolvidos em plágio, mesmo que parcial, terão nota ZERO.