Universidade Federal de Mato Grosso do Sul

Algoritmos e Programação II
Prof. Edson Takashi Matsubara

Trabalho 2
Comparação entre lista estática, lista dinâmica e Hash
Trabalho em dupla
Entrega dia 14/03/2013 via moodle

1. Implemente o TAD lista com as seguintes estruturas:

   (a) lista estática,

   (b) lista encadeada alocada dinâmicamente e;

   (c) uma Tabela Hash (com listas ligadas para tratamento de colisões) com tamanho de 1.000.000 de buckets.

   Vale ressaltar que a Tabela Hash não faz parte do TAD lista. Neste trabalho vamos simular o funcionamento de uma lista com uma hash. O TAD deverá ser implementadas com as seguintes interfaces:

```
void init(lista *l);
int insere(lista * l,tipo val);
int busca(lista *l, tipo val, void * pont);
int remove(lista * l, tipo val);
void stringtipo(void * pont, char ret[]);
```

   Cada TAD deverá ter um arquivo `header` e um arquivo com o código fonte com os seguintes nomes:

   (a) lista estática ordenada (`lista.h` e `lista.c`),

   (b) lista encadeada alocada dinâmicamente ordenada (`listad.h` `listad.c`);

   (c) uma Tabela Hash, com listas encadeadas para tratamento de colisões (`hash.h` e `hash.c`).

2. O presidente da república decidiu fazer uma mega operação nacional para apreender todos os carros roubados do Brasil. Durante uma semana, todos os carros de transitarem por vias públicas serão verificados por meio de blitz em diversos pontos de todas as cidades brasileiras. Você, pelas suas excelentes notas na disciplina de Algoritmos e Programação II da FACOM foi contratado para desenvolver o melhor sistema de busca de placas do Brasil. Implemente um programa capaz de manter uma base de dados atualizada que responde pelos comandos de roubo, apreendido e consulta. A implementação deverá ser feita com uma lista estática. O código fonte deste programa deverá ser chamado de `blits.c` que deverá chamar o `#include "lista.h"`. O executável deverá ter o nome de `blits`.

   Entrada cada linha da entrada corresponde a uma das operações de roubo, apreensão e consulta indicados na entrada pelos caracteres `r`, `a` e `c` respectivamente.

   `r` - roubo (inserção do carro na base de dados). A operação de roubo contém informações da placa do carro, ano, marca, modelo e estado. Estas informações deverão estar armazenadas no registro `tipo`.

```
typedef struct{
   char placa[8];
   int ano;
   char marca[10];
   char modelo[15];
   char estado[3];
}tipo;
```

O comando `r` deve aparecer com os campos na ordem indicada na definição do registro, placa antes de ano, ano antes de marca, marca antes de modelo, modelo antes de estado. Assim a inclusão de um comando de roubo de um carro com a placa HTX8900, ano 2010, marca ford, modelo fiesta e estado MS é dados pela linha:

```
r HTX8900 2010 ford fiesta MS
```

a - apreendido (remoção do carro na base de dados). A apreensão é dada pelo comando `a` seguido da placa do veículo.

```
a HTX8900
```

c - consulta (busca do carro na base). A consulta é dada pelo comando `c` seguido da placa do veículo

```
c HTX8900
```

Um exemplo de entrada é apresentado a seguir:

```
r OFE3457 2012 toyota corolla PB
r MWB8448 1998 gm celta TO
r NRC9060 2011 vw voyage CE
r OEW3785 2012 gm astra PB
a MWB8448
c MWB8448
r OAC4107 2007 ferrari 430scuderia AM
a MWB8448
a MWA1234
r JZA9237 2004 ford ka MT
r MVC5072 2000 gm celta AL
c OMC9815
c HFU5964
c MVC5072
```

Saída Cada comando deve apresentar a sua respectiva saída como descrito a seguir:

r - roubo: a saída deverá fornecer as informações do registro do carro inserido na lista da seguinte maneira:

```
carro (HTX8900 2010 ford fiesta MS) roubado.
```

caso o carro já esteja na lista, o seu programa deverá retornar:

```
carro (HTX8900 2010 ford fiesta MS) ja foi cadastrado.
```

a - apreendido: caso a placa indicada esteja armazenada na lista (banco de dados), o comando de apreendido deverá remover o registro da memória.

```
carro (HTX8900 2010 ford fiesta MS) recuperado.
```

Caso o carro não esteja cadastrado como carro roubado o programa deverá apresentar a seguinte mensagem.

```
carro HTX8900 nao consta na base de dados.
```

c - consulta: caso o carro esteja cadastrado no banco de dados o sistema deverá retornar

```
carro (HTX8900 2010 ford fiesta MS) eh um carro roubado, prender motorista.
```

caso o carro não seja um carro na lista de carros roubados:

```
carro HTX8900 NAO eh um carro roubado, liberar motorista.
```

Segue um exemplo de saída apresentado pela respectiva entrada definida acima

```
carro (OFE3457 2012 toyota corolla PB) roubado.
carro (MWB8448 1998 gm celta TO) roubado.
carro (NRC9060 2011 vw voyage CE) roubado.
carro (OEW3785 2012 gm astra PB) roubado.
carro (MWB8448 1998 gm celta TO) recuperado.
```

```
carro MWB8448 NAO eh um carro roubado, liberar motorista.
carro (OAC4107 2007 ferrari 430scuderia AM) roubado.
carro MWB8448 nao consta na base de dados.
carro MWA1234 nao consta na base de dados.
carro (JZA9237 2004 ford ka MT) roubado.
carro (MVC5072 2000 gm celta AL) roubado.
carro OMC9815 NAO eh um carro roubado, liberar motorista.
carro HFU5964 NAO eh um carro roubado, liberar motorista.
carro (MVC5072 2000 gm celta AL) eh um carro roubado, prender motorista.
```

3. Comparação entre as implementações do TAD. Com a implementação do Item 2 como uma lista estática, troque a linha de `#include "lista.h"` por `#include "listad.h"` produzindo o executável `blitsd`. Faça o mesmo com a hash produzindo o executável `blitsh`.

   Compare o tempo de execução das implementações para as operações de roubo, apreensão e consulta preenchendo a Tabela 1. Juntamente com a descrição do trabalho estão dispostos três arquivos `roubo.txt`, `apreensao.txt` e `consulta.txt` com 1.000.000 de placas para realizar os testes requeridos.

Tabela 1: Tempo de execução

tad	número de carros	roubo	apreensão	consulta
lista estática	1000			
lista estática	5000			
lista estática	10000			
lista estática	50000			
lista estática	100000			
lista estática	500000			
lista estática	1000000			
lista dinâmica	1000			
lista dinâmica	5000			
lista dinâmica	10000			
lista dinâmica	50000			
lista dinâmica	100000			
lista dinâmica	500000			
lista dinâmica	1000000			
hash	1000			
hash	5000			
hash	10000			
hash	50000			
hash	100000			
hash	500000			
hash	1000000			

   Para obter o tempo de apreensão, faça uma entrada concatenando os arquivos de roubo + apreensão e subtraia o tempo de roubo. Faça o mesmo para obter o tempo da consulta.

   Escreva um relatório apresentando os resultados e construa um gráfico que represente a Tabela 1 e descreva as conclusões obtidas.

   O tempo de execução pode ser obtido utilizando a função clock() da biblioteca time. Segue um exemplo que retorna esta informação.

```
#include <stdio.h>
#include <time.h>

int main(void){
    clock_t start = clock();
    int i;
    int sum = 0;
    for (i=0;i<1000000000;i++){
        sum += 1;
    }
```

```
    printf("Time elapsed: %f\n", ((double)clock() - start) / CLOCKS_PER_SEC);
}
```

Não esqueça da conversão para double.