UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO
http://www.icmc.usp.br/

Professores: Fernando V. Paulovich (paulovic at icmc.usp.br)
Moacir Ponti Jr (moacir at icmc.usp.br)
Aluno PAE: Felipe S. L. G. Duarte (fgduarte at icmc.usp.br) - Turma A
Monitor: Cassiano K. Casagrande (cassianokc at usp.com) - Turma B

Trabalho 04: L TUSCA

1 Prazos, Especificações e Observações importantes:

- O trabalho descrito a seguir é individual e não será tolerado qualquer tipo de plágio ou cópia em partes ou totalidade do código. Caso seja detectado alguma irregularidade, os envolvidos serão chamados para conversar com o professor responsável pela disciplina.
- A entrega deverá ser feita única e exclusivamente por meio do Sistema de Submissão de Programas (SSP) no endereço eletrônico`http://ssp.icmc.usp.br` até o dia **30 de junho de 2013 as 23 horas e 59 minutos**. Sejam responsáveis com o prazo final para entrega, o SSP está programado para não aceitar submissões após este horário e não será aceito entrega fora do sistema.
- A interpretação desta descrição faz parte do trabalho. Leia a descrição do trabalho com atenção e várias vezes, anotando os pontos principais e as possíveis formas de resolver o problema. Comece a trabalhar o quanto antes para não ficar dúvidas e você consiga entregar o trabalho a tempo.
- O trabalho deverá ser submetido em formato zip/Makefile contendo o arquivo principal, o Makefile, as TAD's implementadas (pares `.h .c`) e um arquivo texto com o nome e número USP. Serão avaliados não só o resultado final do sistema, mas também a identação, modularização e documentação interna.
- A implementação é livre, crie quantas TAD's julgarem necessárias. Importante notar que a modularização do código bem como a forma como as TAD's foram criadas serão levados em consideração na atribuição final da nota.
- Compile o programa em um sistema operacional linux antes de submeter ao SSP, **trabalho em que o comando** `make` **e** `make`

`run` **não funcione, não será corrigido e consequentemente receberá nota zero**. Caso não tenha o sistema operacional linux instalado em sua maquina, aconselha-se a utilização de uma máquina virtual. Como instalar o sistema operacional linux em uma maquina virtual está descrito no seguinte tutorial: http://maistutoriais.com/2012/01/instalar-maquina-virtual-usando-o-virtual-box/.

- Referencie, com um comentário no próprio código, qualquer algoritmo ou trecho de código retirado da internet. Código copiado sem a devida referencia é considerado plágio que por sua vez é crime.
- Nenhuma saída do trabalho será em arquivo, ou seja, o resultado dos algoritmos deverá ser exibidos utilizando o stdout.

2 Descrição do Problema:

A L Taça Universitária de São Carlos (TUSCA) está chegando!!! Estamos no ano de 2029 e o evento será realizado no segundo semestre do ano, os organizadores já estão se apressando para darem conta de todo o serviço.

A TUSCA já foi a maior festa da cidade, hoje reune cerca de 5.000 pessoas (Sim, depois de varias liminares da justiça e proibições de diversos órgãos públicos, eles "quase" conseguiram acabar com o TUSCA). O torneio, que acontece anualmente e tem duração de quatro dias, reúne as duas grandes universidades da cidade: USP e UFSCar. Outras faculdades convidadas participam do torneio e são escolhidas pela organização anualmente.

Acreditem ou não, o objetivo da TUSCA é somente promover uma competição poliesportiva entre as universidade participantes (isso porque as festas estão proibidas desde 2011). O sistema que você havia feito em 2013 começou a apresentar lentidão nas buscas devido o tamanho dos arquivos de índice e dados.

Você foi chamado para fazer uma manutenção provisória para que o sistema volte a ter o desempenho de sempre. Para isso, os organizadores precisarão que o sistema continue capaz de cadastrar todos os participantes para ter um maior controle de todos que estão inscritos como atletas. Sendo um excelente programador formado pela USP, você lembrou da arvore B e de todo o seu potencial. Este é o seu trabalho, troque o sistema de índices feito no trabalho três pela indexação utilizando arvore B, o sistema ainda deve ser capaz de cadastrar as informações de cada atleta (CPF, Nome, Registro Acadêmico, Universidade e Modalidade Esportiva) e efetuar buscas posteriormente.

3 Organização de Arquivos

Para realização deste trabalho será necessário criar e manipular somente 2 arquivos distintos, são eles:

1. `'data.db'` - Arquivo contendo os dados de atletas inscritos. Tem sua arquitetura com registros com tamanho fixo. O delimitador entre os caracteres será a barra vertical '`|`'. Este caractere, garantidamente, não irá aparecer em nenhum dos campos. Cada campo de registro do atleta também terá tamanho fixo:
    a. `'CPF'` - **Chave Primária** Este campo deverá conter o número do CPF de cada atleta (máximo 11 caractere).
    b. `'Nome'` - Este campo terá o nome do aluno (máximo 30 caracteres).
    c. `'Registro Acadêmico'` - Este campo terá o registro acadêmico de cada atleta (máximo 10 caractere).
    d. `'Universidade'` - **Chave Secundária** - Este campo terá o nome da universidade que o atleta estuda (máximo 30 caracteres).
    e. `'Modalidade Esportiva'` - **Chave Secundária** - Este campo terá o esporte que o atleta está inscrito (máximo 30 caracteres).

    Assim, o arquivo de dados terá a seguinte estrutura:

```
19088419718|Felipe_Duarte                  |6426830
    |USP_CAASO                       |Basquete_Masc
ulino            |
19263473718|Karina_Bindandi               |67840
34   |USP_CAASO                        |Volei_Femin
ino                |
**********|Rafal_Clever                  |18732
_
02  |UNICAMP                           |Truco_Mascul
ino              |
21371226539|cassiano_Casagrande           |21736
12   |USP_CAASO                        |Xadrez_Masc
ulino            |
```

2. `'prim.idx'` - Arquivo contendo a arvore-B indexando a relação chave primaria / posição do registro no arquivo. A arvoré-B deve, necessariamente ter **ordem 4**, ou seja, armazena 3 chaves e 4 ponteiros por pagina.

Leia atentamente todo o material dado em sala de aula sobre a teoria de chaves primárias, secundárias e lista invertida. É estritamente importante que você conheça a teoria para saber o conteúdo e a estrutura de cada arquivo acima.

4 Comandos do Sistema:

- `'sair'` - Este comando deverá sair do sistema lib rando toda memória possivelmente alocada e fechando corretamente possíveis arquivos abertos;
- `'cadastrar'` - Este comando deverá cadastrar um atleta com todas as suas informações. Nenhuma informação irá conter espaço em branco entre suas palavras. Os dados de cada atleta serão informados na seguinte ordem:
    1. `'CPF'` - máximo 11 caractere.
    2. `'Nome'` - máximo 30 caracteres.
    3. `'Registro Acadêmico'` - máximo 10 caractere.
    4. `'Universidade'` - máximo 30 caracteres.
    5. `'Modalidade Esportiva'` - máximo 30 caracteres.

    Importante ressaltar que de acordo com a teoria de chaves primarias, não pode existir dois registros na base de dados com a mesma chave primaria. Assim, após o usuário informar os dados do atleta, o sistema deverá verificar a existência de uma registro com a mesma chave primária. Caso não exista a inserção deve ser efetuada normalmente. Caso já exista um registro com aquela chave primaria, o sistema não deverá efetuar o cadastro e deverá exibir a seguinte mensagem de alerta na tela

```
printf("Conflito de chave primaria. Registro nao i
  nserido!\n");
```

- `'buscar <cpf>'` - Este comando deverá buscar os dados do atleta de acordo com a chave primária. Para isso, **obrigatoriamente**, deve-se usar árvore-B e toda a estrutura de indexação que ela proporciona. Os dados dos atletas devem ser impressos na tela da seguinte maneira:

```
printf("%s - %s\n", CPF, Nome);
printf("\tRegistro Academico: %s\n", registroAca
demico);
printf("\tUniversidade: %s\n", universidade);
printf("\tModalidade: %s\n", modalidade);
```

- ‘dump <nome do arquivo>’ - Este comando deverá imprimir o conteúdo de cada arquivo da seguinte maneira:
    1. ‘dump data.db’ - Este comando deverá imprimir na tela todo o conteúdo do arquivo data.db (char-por-char). Ao fim, coloque um ‘\n’ final.
    2. ‘dump prim.idx’ - Este comando deverá imprimir na tela a estrutura da arvore-B armazenada no arquivo prim.idx. Deve imprimir a árvore usando um algoritmo parecido com “Pré-Ordem” disponível a seguir.

```
void printBTree(int rrn, int altura){
  if(rrn == NIL) return;

  int i;
  no_arvore* nextPage = carrega_pagina(rrn);

  printf("Altura: %d | num. Chaves: %d | chaves
= [ ", altura, nextPage->num_chaves);
  for (i = 0; i < nextPage->num_chaves; i++) {
    printf("%s ", nextPage->chaves[i]);
  }
  printf("]\n");
  for (i = 0; i < ORDEM; i++) {
    printBTree(nextPage-
>filhos[i], altura + 1);
  }
}
```