BCC - EDA2 - Prova 1 - 2 chamada

Prova 1 da disciplina EDA2 (Estrutura de Dados II). O formulário estará disponível para preenchimento até as 23:59 do dia 21/04.

* Indica uma pergunta obrigatória

E-mail *

leandro.rittes1990@gmail.com

Nome completo *

Leandro Ribeiro Rittes

1) Sobre arquivos indexados, é correto afirmar: (1,0 ponto) *

I. Suportam múltiplos índices.
II. Deve possuir ao menos um índice exaustivo.
III. Deve possuir ao menos um índice seletivo.
IV. Cada índice contém uma réplica dos registros para melhorar o desempenho.
V. Não requer chave primária para identificação dos registros.

- (●) Somente I e II estão corretas.
- ( ) Somente I, II e III estão corretas.
- ( ) Somente I, II e V estão corretas.
- ( ) Somente II, IV e V estão corretas.
- ( ) Somente III, IV e V estão corretas.

2) Sobre arquivos diretos, é correto afirmar: (1,0 ponto) *

I. Fornece um acesso rápido aos registros ao percorrer uma estrutura auxiliar baseada em índice.
II. Usa uma função *hashing* para calcular o endereço do registro pela chave de acesso.
III. Funções *hashing* sempre geram um único endereço para cada entrada.
IV. Funções *hashing* podem gerar colisões de endereços em uma mesma entrada.
V. No arquivo direto não são previstos acessos seriais (sequenciais).

- ○ Somente I está correta.
- ○ Somente I e V estão corretas.
- ○ Somente II e III estão corretas.
- ○ Somente II, III e IV estão corretas.
- ◉ Somente II, IV e V estão corretas.

3) As métricas para avaliação de desempenho da memória cache são: (1,0 ponto) *

I. *Hit*: dado encontrado no nível procurado.
II. *Miss*: dado não encontrado no nível procurado.
III. *Hit-time*: tempo de acesso considerando o tempo para verificar se é *hit* ou *miss*.
IV. *Miss-penalty*: tempo para transferir dados de níveis baixos, quando não encontrado.
V. *Hit-rate* e *miss-rate*: percentual de *hits* e *misses* no nível.

- ○ Somente I e II estão corretas.
- ○ Somente II e III estão corretas.
- ○ Somente III e IV estão corretas.
- ○ Somente IV e V estão corretas.
- ◉ Todas as alternativas estão corretas.

4) Sobre a classificação externa de arquivos, é correto afirmar: (1,0 ponto)

I. A classificação externa utiliza exclusivamente a memória principal.
II. A estratégia de classificação externa contém duas fases: classificação e intercalação.
III. A seleção por substituição utiliza marcações (congelados e não congelados) sobre os registros para se beneficiar de uma possível classificação parcial do arquivo.
IV. O objetivo da intercalação consiste em transformar um conjunto de partições classificadas em um único arquivo ou partição.
V. O objetivo da intercalação é aumentar o número partições ao dividir as partições já classificadas, tornando possível manipulá-las em memória principal.

- ○ Somente I e II estão corretas.
- ○ Somente I, II e III estão corretas.
- ○ Somente II e III estão corretas.
- ◉ Somente II, III e IV estão corretas.
- ○ Somente II, III e V estão corretas.

Limpar seleção

5) Pangrama é uma frase em que são usadas todas as letras do alfabeto de um dado idioma. Faça um programa que leia uma frase e exiba verdadeiro se a frase em questão é um pangrama para um alfabeto de 'a' a 'z', incluindo 'k', 'w' e 'y' e desconsiderando acentuações. Caso seja um pangrama, o também deve ser retornado se o pangrama é perfeito ou não, isto é, cada letra do alfabeto aparece uma única vez. (3,0 pontos) *

Exemplo
Entrada: "Bancos futeis pagavam-lhe queijo, whisky e xadrez"
Saída = Pangrama verdadeiro e imperfeito.

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

int main()
{
   int letters[26] = {'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y',
'z'};
   int finded[26] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};

   char sentence[1000];
   scanf("%[^\n]", sentence);

   int isUnique = 1;

   for (int i = 0; sentence[i] != '\0'; i++)
   {
      for (int j = 0; j < 26; j++)
      {
         if (tolower(sentence[i]) == letters[j])
         {
            finded[j]++;
         }
      }
   }

   for (int j = 0; j < 26; j++)
   {
      if (finded[j] == 0)
      {
         printf("Não é um pangrama!\n");
         return 0;
      }
      if (finded[j] > 1)
      {
         isUnique = 0;
      }
   }

   printf("Pangrama verdadeiro e ");
   if (isUnique == 1)
   {
      printf("Perfeito");
   }
   else
   {
      printf("Imperfeito");
   }
   printf("\n");

   return 0;
}
```

6) Distância de edição é uma medida para indicar o quão próximas são duas strings. Esta medida é calculada a partir do número mínimo de operações necessárias para transformar uma string na outra. As operações válidas são: inserção, deleção ou substituição de um caractere. Faça um programa que leia duas cadeias de caracteres e, em seguida, o programa deve retornar o número de operações necessárias para transformar a primeira cadeia na segunda. A solução proposta deve implementar o algoritmo de Levenshtein. (3,0 pontos) *

Observação: dependendo do SO/encoding utilizado, o exemplo pode variar de 4 a 6 devido a acentuação/caracteres especiais.

- Exemplo, dadas as *strings* `"exercício"` e `"exército"`, a saída esperada é 4.
  - `"exercício"`
  - `"exército"`
  - `"__s__ssr" //_: igual, s: substituição, r: remoção`

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

int min(int val1, int val2, int val3)
{
   int menor = val1;
   if (val2 < menor)
      menor = val2;
   if (val3 < menor)
      menor = val3;
   return menor;
}

int dist_levenshtein(char *str1, char *str2)
{
   int m = strlen(str1);
   int n = strlen(str2);

   int dp[m + 1][n + 1];

   for (int i = 0; i <= m; i++)
   {
      dp[i][0] = i;
   }

   for (int j = 0; j <= n; j++)
   {
      dp[0][j] = j;
   }

   for (int i = 1; i <= m; i++)
   {
      for (int j = 1; j <= n; j++)
      {
         if (str1[i - 1] == str2[j - 1])
         {
            dp[i][j] = dp[i - 1][j - 1];
         }
         else
         {
            dp[i][j] = min(dp[i - 1][j], dp[i][j - 1], dp[i - 1][j - 1]) + 1;
         }
      }
   }

   return dp[m][n];
}

int main()
{
   char str1[100];
   char str2[100];
```

```
    char str2[100];

    printf("Digite a primeira string: ");
    scanf("%s", str1);

    printf("Digite a segunda string: ");
    scanf("%s", str2);

    int dist = dist_levenshtein(str1, str2);

    printf("Distância de edição: %d\n", dist);

    return 0;
}
```