P3 Programação II 2013.1 Departamento de Informática/PUC-Rio

27 de junho de 2013

Aluno: _____

Matrícula:_____ Turma: _____

	Valor	Nota
Q1	3.0	
Q2	3.0	
Q3	4.0	
Total	10.0	

Declaro ter lido as instruções abaixo e estar ciente das normas da aplicação da Prova.

Assinatura: _____

INSTRUÇÕES:

1. A prova é sem consulta e sem perguntas. A interpretação do enunciado faz parte da prova.
2. A prova deve ser completamente resolvida nas folhas que constam deste caderno, utilizando-se frente e/ou verso;
3. A prova pode ser feita utilizando-se lápis ou caneta (azul ou preta);
4. Dispositivos eletrônicos (celulares, tablets, …) devem ser desligados.
5. **O aluno só pode manter junto a si lápis, borracha e caneta. Todo papel (independentemente do conteúdo) ou dispositivo (celular, tablet, etc) (ligado ou não) encontrado junto ao aluno implicará o recolhimento**

stdio.h:

```
int scanf (char* formato, ...);
int printf (char* formato, ...);
FILE* fopen (char* nome, char* modo);
int fclose (FILE* fp);
int fscanf (FILE* fp, char* formato, ...);
int fprintf (FILE* fp, char* formato, ...);
char*fgets(char* str, int size, FILE* fp);
int sscanf(char* str, char* formato, ...);
```

math.h:

```
double sqrt (double x);
double pow (double x, double exp);
double cos (double radianos);
double sin (double radianos);
```

string.h:

```
int strlen (char* s);
int strcmp (char* s, char *t);
char* strcpy (char* destino, char* fonte);
char* strcat (char* destino, char* fonte);
char* strdup (char* s);
```

stdlib.h:

```
void qsort (void *v, int n, int tam, int (*cmp)(const void*, const void*));
void* malloc (int nbytes);
void free (void* p);
```

QUESTÃO 1:

Uma mensagem foi armazenada em uma cadeia de caracteres de forma codificada. A codificação foi feita invertendo-se as subsequências de não vogais.

Assim, a mensagem que originalmente era no exemplo 1:

"PROGRAMA DA PROVA" , foi armazenada da seguinte maneira:

"RPORGAMAD ARP OVA"

E a mensagem que originalmente era no exemplo 2:

"ATRASOS PREVISTOS NOS PROJETOS", foi armazenada da seguinte maneira:

"ARTASORP SEVITSON SORP SOJETOS" .

Usando obrigatoriamente uma pilha como estrutura auxiliar, escreva a função **decifra**, que recebe (o ponteiro para) uma cadeia contendo uma mensagem codificada e exibe a mensagem original. (É só para exibir.)

Assuma que todo caracter correspondente a uma letra do alfabeto está em maiúsculo.

(Observe que branco/espaço é uma "não-vogal") .

Para isso deve ser utilizada uma pilha de caracteres. Considere a existência de um TAD Pilha de caracteres (Pilha de char), que tem o seguinte arquivo de interface:

```
/* arquivo PilhaDeChar.h   */
/* tipo e funções básicas */

typedef struct pilhaChar PilhaChar;

/* função que cria, inicializa e retorna (o endereço de) uma pilha de caracteres  */
PilhaChar *pilha_cria(void);

/* função que recebe (o endereço de) uma pilha de caracteres  e um caracter,
inserindo-o na pilha  */
void pilha_push (PilhaChar *p, char novo);

/* função que recebe (o endereço de) uma pilha de caracteres e faz uma retirada da
pilha, retornando o caracter do topo da pilha, ou seja o mais recentemente inserido */
char pilha_pop(PilhaChar *p);

/* função que recebe (o endereço de) uma pilha de caracteres e retorna 1, se a pilha
está vazia, ou 0, em caso contrario */
int pilha_vazia(PilhaChar *p);

/* função que recebe (o endereço de) uma pilha de caracteres e libera o espaço ocupado
pela pilha */
void pilha_libera(PilhaChar *p);
```

Obs: Não esqueça de liberar o espaço ocupado pela pilha quando ela não for mais necessária.

SOLUÇÃO DA Q1:

Continuação da solução da questão1:

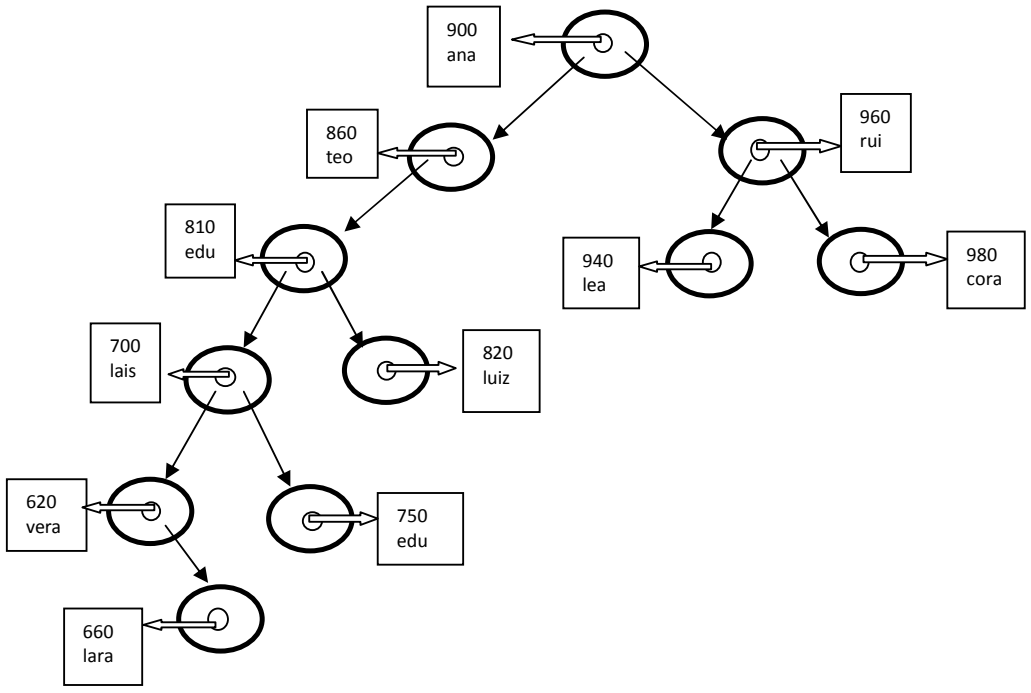
QUESTÃO 2:
Considere o tipo estruturado Candidato:

```
struct candidato
{
   int inscricao;
   char nome[51];
};
typedef struct candidato Candidato;
```

Considere o seguinte nó de uma árvore binária de busca (ABB):

```
struct noArv
{  Candidato    * info;
   struct noArv * esq;
   struct noArv * dir;
};
typedef struct noArv NoArv;
```

Considere também que a árvore binária de busca está ordenada em ordem crescente de inscrição. Escreva a função recursiva **exibeNomes** que recebe a raiz da árvore, uma certa inscrição X e um nome; e "levando em conta a ordenação da árvore", imprime todos os nomes dos candidatos que têm número de inscrição menor ou igual à inscrição X, desde que o nome do candidato seja diferente do nome recebido. Por exemplo, no caso da árvore abaixo, se fizéssemos a chamada fornecendo a raiz da árvore, o valor de inscrição 880 e o nome "edu" teríamos impressos os seguintes nomes: lara, vera, lais, teo e luiz. Observe que os nomes podem aparecer em qualquer ordem, desde que atendam ao critério descrito (inscrição <= x e nome do candidato != nome recebido). Mas ao percorrer a árvore binária de busca para fazer tal exibição tem que ser levada em conta a sua ordenação.



Você pode usar a função "strcmp" da biblioteca do C.

SOLUÇÃO DA Q2:

QUESTÃO 3:
Considere a existência de duas listas simplesmente encadeadas, ORDENADAS crescentemente por **codigo**, em que cada nó encadeado é representado pelo tipo NoLista abaixo.

```
typedef struct noLista  NoLista;
struct noLista
{   int     codigo;
    char   nome[51];
    NoLista *prox;
};
```

Escreva a função **eliminaNos** que recebe duas listas simplesmente encadeadas (ou seja, os endereços dos primeiros nós das duas listas) e ordenadas crescentemente por "codigo" e retorna o endereço do primeiro nó da lista 1 cujos nós com valores iguais aos encontrados na lista 2 foram eliminados. NÃO USE FUNÇÕES AUXILIARES (exceto as disponibilizadas pelas bibliotecas padrão de C) e LEVE EM CONSIDERAÇÃO A ORDENAÇÃO DAS LISTAS para tornar sua função mais eficiente.

Por exemplo, se os elementos da lista 1 fossem:
[111,”leo”], [333, “rui”], [446, “cris”], [558, “lara”], [666,”cadu”], [777,”rosa”], [999,”bia”]

E os elementos da lista 2 fossem:
[111,”leo”], [333, “rui”], [440, “edu”], [554, “vera”], [777,”rosa”], [888,”lia”]

A lista 1 resultante teria os seguintes valores:
[446, “cris”], [558, “lara”], [666,”cadu”], [999,”bia”]

e seria retornado o endereço do nó com [446, “cris”].

Note que a lista 1 alterada continua ordenada em ordem crescente de "codigo". Considere que nas listas recebidas, nem todos os valores da lista 2 estão na lista 1 e vice-versa. Considere também que não há repetições de "codigo" numa mesma lista e que um determinado "codigo" corresponderá sempre a um mesmo nome.

SOLUÇÃO DA QUESTÃO 3:

Continuação da solução da questão 3: