



Instituto Federal do Espírito Santo

Campus Serra

Coordenadoria de Informática

Curso Superior Bacharelado em Sistemas de Informação

4ª Prova de Programação 2

Semestre 2015-1

Leia com atenção o enunciado a seguir e faça o que é pedido e da forma como é pedido.

Preste atenção nos requisitos abaixo e não tire zero pontos por não segui-los.

Requisitos:

- 1. Não basta que o programa funcione. A construção dos programas deve obedecer aos requisitos e procedimento e padrões vistos em aula e laboratório.**
- 2. Todas as funções devem ser construídas usando-se o princípio da economia. Somente serão aceitas funções auxiliares se elas fatorarem algum tipo de código repetitivo (usando por mais de uma das funções pedidas). Caso contrário, a questão sofrerá descontos.**
- 3. Só serão aceitas soluções construídas a partir dos conteúdos vistos em aula, e nada mais. Nenhuma biblioteca matemática ou qualquer comando que substitua códigos pedidos nos enunciados.**
- 4. Cópias receberão a nota zero. O que serão consideradas cópias ??**

Resposta: Os programas serão submetidos a uma ferramenta computacional de aferição de similaridade. Códigos que alcançam índice de similaridade $\geq 75\%$ serão considerados cópias e terão a pontuação zerada.

O professor se reserva o direito de convidar qualquer aluno a prestar qualquer esclarecimentos sobre o código presente nas respostas da avaliação, e esses esclarecimentos farão parte da avaliação.

- 5. Irão produzir descontos na pontuação, além do não cumprimento dos requisitos:**
 - **Nomes esdrúxulos de variáveis;**
 - **Variáveis que mudam de tipo ao longo do programa;**
 - **Nomes que remetem a um contexto estranho ao uso e/ou estrutura da variável;**
 - **Construções de controle que não sejam as utilizadas em aula;**
 - **Falta de organização na construção dos fontes.**

Ao iniciar a confecção da prova o aluno estará sujeito a todas essas regras de avaliação.

Questão 1 (15 pontos)

A criptografia é um conjunto de técnicas de codificação que previne o acesso não autorizado à mensagens consideradas confidenciais. Existem vários métodos para criptografar mensagens.

No Método de César, cada letra do alfabeto é substituída por outra transladada algumas posições à frente. A chave 'secreta' (informação que permite codificar e decodificar a mensagem) é a quantidade de posições deslocadas. Esse deslocamento é muitas vezes chamado de rotacionamento, porque é cíclico. Ou seja, deslocar a última letra 'Z' à 1 posição direita, obtêm-se a letra 'A'. Ver figura 1.a para detalhes. Nas implementações computacionais, a tabela de caracteres (ASCII, UTF-8, etc) é utilizada no lugar do alfabeto contendo apenas letras.

Já no Método da Tabela Espartana (aqui adaptado para a prova), distribui-se a mensagem na menor matriz quadrada capaz de contê-la. Se a menor matriz quadrada calculada possui mais espaços que o tamanho da mensagem, insere-se brancos na mensagem até completar o mesmo tamanho da matriz.

Observe que os brancos deve ser inseridos ao longo da mensagem/texto e não somente ao seu final. Ou seja, deve-se aumentar a quantidade de brancos entre as palavras acrescentando novos brancos à frente dos já existentes. No próximo passo, distribui-se sequencialmente a mensagem pelos elementos da matriz. Concluída a distribuição, a matriz é percorrida segundo um trajeto específico. No nosso caso, na sequencia diagonal secundária inversa (de baixo para cima), como ilustrado na figura 1.b. A mensagem criptografa é justamente o texto formado pelas letras encontradas ao longo desse trajeto. A chave 'secreta' (informação que permite codificar e decodificar a mensagem) é justamente as dimensões da matriz quadrada mais o percurso usado para percorrê-la.

Construa o que é pedido obedecendo aos requisitos citados:

a) Função *cifraCesar(<mensagem>,<ndesloc>)*: criptografa uma mensagem passada como parâmetro de entrada, tendo como chave de criptografia é o deslocamento passado como segundo parâmetro. A função devolve como valor de retorno a mensagem criptografada segundo o Método de César descrito anteriormente. Como requisito obrigatório, a função deve ser autocontida, ou seja, deve fazer a criptografia por completo, sem recorrer a nenhuma outra função auxiliar. (1 ponto)

b) Função *decifraCesar(<mensagem>,<ndesloc>)*: descriptografa uma mensagem codificada segundo o Método de César passada como parâmetro de entrada. A chave de criptografia é o deslocamento passado como segundo parâmetro. A função devolve como valor de retorno a mensagem original, antes de ser criptografa. Como requisito obrigatório, a função deve ser autocontida, ou seja, deve fazer a criptografia por completo, sem recorrer a nenhuma outra função auxiliar. (1 ponto)

Requisito adicional para a) e b): as funções devem implementar o deslocamento utilizando a codificação (ASCII, UTF-8, etc.) dos caracteres que compõem a mensagem.

c) Função *cifraEspartana(<mensagem>)*: criptografa uma mensagem passada como parâmetro de entrada utilizando o método de Criptografia Espartana descrito anteriormente. A função deve: i) avaliar corretamente o tamanho apropriado para a matriz de criptografia; ii) se for o necessário, complementar a mensagem original com o caractere @ até que a mensagem assuma o tamanho adequado para ser corretamente distribuída na matriz de criptografia. O caractere @ faz o papel de brancos adicionais mencionado na descrição anterior do método. Lembre-se que 'brancos' adicionais (@) só são aceitos no final da mensagem quando esta não possui nenhum branco no seu conteúdo, justamente o caso em que a mensagem é formada por apenas 1 palavra. Somente nesses casos é permitido acrescentar os 'brancos' adicionais somente ao final da mensagem; iii) percorrer a matriz de acordo com a direção mostrada na figura e criar a mensagem criptografada. (5 pontos)

Crie funções auxiliares para desempenhar as tarefas i) e ii).

d) Função *decifraEspartana(<mensagem>)*: descriptografa uma mensagem criptografada pelo método de Criptografia Espartana descrito anteriormente. A função deve proceder o processo inverso ao da criptografia em *cifraEspartana(<mensagem>)*. Como retorno, a função deve devolver a mensagem original. Lembrar que a mensagem resgatada conterá 'espaços' adicionais na forma de caracteres @. Esses caracteres devem ser removidos no último passo executado pela função antes do retorno do conteúdo descriptografado. (5 pontos)

e) Função *criptografa(<mensagem>)*: esta função efetua a criptografia do parâmetro texto de entrada utilizando os métodos implementados nos itens a) e c). Na função, a mensagem deve primeiramente ser criptografada pelo Método de César (a)) com deslocamento 13. Em seguida, o produto dessa criptografia deve ser fornecido como entrada para ser novamente criptografado pelo Método Espartano (c)). O resultado final deve ser retornado pela função. (1.5 pontos)

f) Função *descriptografa(<mensagem>)*: esta função efetua a decodificação do parâmetro texto de entrada utilizando os métodos implementados nos itens b) e d). Na função, a mensagem deve primeiramente ser decodificada pelo Método de César (b)) com deslocamento 13. Em seguida, o produto dessa decodificação deve ser fornecido como entrada para ser novamente decodificado pelo Método Espartano (d)). O resultado final deve ser retornado pela função, a mensagem original. (1.5 pontos)

Construa uma aplicação de testes para avaliar a sua implementação dos itens a) a f). Para fins de teste, utilize o arquivo *cripto.txt*, fornecido pelo professor. O arquivo contém o equivalente criptografado (por meio de e)) da mensagem de autoria do Mestre Yoda:

“Em um lugar escuro nos encontramos e um pouco mais de conhecimento ilumina nosso caminho”

Utilize o arquivo para testar tanto a descriptografia quanto a criptografia (descriptografe, criptografe, descriptografe, e terá a mensagem original como retorno).

Na correção, o professor utilizará para teste outros arquivos do *cripto.txt*.

Questão 2 (20 pontos)

Um arquivo texto contém os dados de várias matrizes na forma de descrições textuais. No arquivo, cada linha descreve completamente uma matriz. A sequência de conteúdo é a da esquerda para direita, cima para baixo, como corre na leitura de uma matriz. O arquivo foi gerado por uma ferramenta assistiva de captura de voz. Essa ferramenta transcreve a fala do usuário, formata o dado e gera o arquivo. A quantidade de matrizes no arquivo é indeterminada, assim como as dimensões das matrizes. A estrutura do arquivo pode ser compreendida a partir da análise da amostra exibida abaixo (arquivo com apenas duas matrizes descritas textualmente):

```
quatro por quatro;duzentos e setenta e dois;novecentos e sessenta e
seis;trezentos e setenta e quatro;mil e duzentos e vinte e sete;trezentos e
sessenta e sete;oitocentos e vinte e um;quatrocentos e cinquenta e oito;mil e
oitocentos e dois;seis;setecentos e cinquenta e dois;setecentos e noventa e
quatro;mil e quatrocentos e noventa e sete;mil e duzentos e cinquenta e
tres;quinhentos;trezentos e sessenta;mil e quarenta e seis;
.....
dois por dois;mil e seiscentos e quarenta e quatro;trezentos e setenta e
oito;mil e quinhentos e cinquenta e seis;mil e setecentos e quarenta;
```

Obs.: a linha pontilhada é apenas para facilitar a visualização na documentação.

Construa o que é pedido obedecendo aos requisitos citados:

- Função `txt2int(<texto com num int>)`: A função deve receber como entrada um texto descrevendo um número inteiro entre 0 e 1999. A função devolve como valor de retorno equivalente int numérico do texto. Como requisito, a função deve ser autocontida e resolver o problema sem a necessidade de recorrer a funções auxiliares. (2.5 pontos)
- Função `txt2mat(<descrição textual da matriz>)`: função que recebe como parâmetro de entrada a descrição textual de uma matriz (ver amostra do arquivo acima) e retorna o equivalente numérico da matriz (utilizando listas). (2.5 pontos)
- Função `det(<param matriz inteiros>)`: calcula e retorna o valor do determinante da matriz quadrada de inteiros passada como parâmetro de entrada. A implementação utilizada deve ser **recursiva** e utilizar o algoritmo de LaPlace para efetuar os cálculo. (10 pontos)
- Utilizando como funções não nativas do python apenas as funções construídas nos itens a) a c), construa uma aplicação que gera um arquivo de índices de um arquivo texto contendo descrição textuais de matrizes, a exemplo do que foi descrito no paragrafo inicial da questão. A aplicação só estará correta se gerar o arquivo de índices corretamente e processar o arquivo de entrada como especificado. Confira o produto da sua implementação usando os arquivos `segredoq2.txt` e `segredoq2.ndx`, fornecidos pelo professor. (5 pontos)

O arquivo de índices é formado por linhas contendo, cada uma, os seguintes pares de informação: *determinante da matriz, linha onde a matriz reside no arquivo de matrizes*. Segue abaixo o arquivo de índices gerado para o arquivo de matrizes exibido no início da questão. O arquivo de índices deve conter o mesmo nome do arquivo de dados, mas com a extensão `ndx`.

```
326962207332, 0
2272392, 430
```

Para testar a sua aplicação, leia o arquivo de índices gerado, acesse um registro qualquer desse arquivo, encontre a linha com a matriz correspondente no arquivo de descrição textual. Em seguida, converta a matriz para a sua forma numérica, calcule o seu determinante e compare com a entrada correspondente no arquivo de índices. Se forem iguais, então o seu programa estará funcionando como esperado. Esse será o procedimento utilizado pelo professor para a correção, além da verificação da estrutura e qualidade do programa.

Como material fornecido para a prova, constam dois arquivos texto. Um arquivo chamado *egredoq2.txt*, contendo duas matrizes textuais (início da questão), e o arquivo *segredoq2.ndx*, contendo os respectivos índices do arquivo de dados (pares determinante, posição do arquivo de dados). Utilize esses arquivos para fazer os seus testes.

Figura 1 (para questão 1)

1.a. Método de César

Nesta cifra cada letra do alfabeto é substituída por outra transladada algumas posições à frente. (chave \leftrightarrow n.º de posições).

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	X	Y	Z	*	,	.	:	:
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	X	Y	Z	_	,	.	:	:	A	B	C

Obs importante: para fins de visualização apenas, * (asterisco) representa espaço em branco.

Exemplo:

Mensagem Original: AULA DE MATEMÁTICA Chave Deslocamento = 3

Mensagem Criptografada: DXOD*GH*PDWHPDWLFD

1.b. Método da Tabela Espartana

Exemplo:

Mensagem Original: AULA DE MATEMÁTICA

Tamanho: 18 letras

Próxima matriz quadrada que cabe 18 conteúdos: 5 x 5 (25 posições)

Mensagem com brancos inseridos: AULA@@@@*DE@@@@*MATEMÁTICA

Mensagem distribuída na matriz e o percurso de diagonal inversa:

A	U	L	A	@
@	@	@	*	D
E	@	@	@	
M	A	T	E	M
A	T	I	C	A

Mensagem Criptografada: A@UE@LM@A@A@**@T@DIE*CMA

Obs: os dados são apenas para ilustração. Lembrando que @ são 'espaços inseridos' para que a mensagem alcance o tamanho de uma matriz quadrada.

Obs: todos os dados de apoio e testes foram construídos e realizados no sistema operacional Linux Ubuntu 12.x. .

Dúvidas ? Contacte o professor.

Boa Prova!