MAC0110 – Introdução à Ciência da Computação
Bacharelado em Ciência da Computação
IME – Primeiro Semestre de 2015

Primeiro Exercício-Programa 3 (EP3)
Professor: André Fujita

Data de entrega: até 23:55 do dia 31 de maio de 2015.

Enigma

Neste EP vamos brincar de Alan Turing e quebrar o Enigma versão "Beginners". Quem ainda não assistiu o filme "O Jogo da Imitação", recomendo fortemente.
(http://pt.wikipedia.org/wiki/O_Jogo_da_Imita%C3%A7%C3%A3o)

Estamos no ano de 2015 onde existe uma grande luta para ser aprovado em MAC110. As notas da P1 foram terríveis, mas uma fonte acaba de informar que as questões das provas são colocadas sistematicamente 24 horas antes do início da prova na internet. O problema é que as questões se encontram criptografadas. Então, para salvar a turma de MAC110 de uma possível reprovação em massa, é imprescindível quebrar o código usado pelo professor e resolver as questões antes da prova.

Após inúmeras tentativas e vidas de veteranos sacrificadas, finalmente se conseguiu obter um exemplar da máquina ENIGMA usada pelo professor. Analisando a máquina, percebeu-se que o algoritmo usado é o Twist descrito na seção seguinte. Sua tarefa: escrever um programa que decripta uma mensagem encriptada pelo método Twist. Boa sorte. Faltam menos de 24 dias para a entrega do EP3.

1. Método Twist

Criptografia é uma área da Ciência da Computação que estuda os métodos de comunicação secreta que transformam uma mensagem (***plaintext***) em uma mensagem cifrada (***ciphertext***), de forma que apenas o real receptor da mensagem seja capaz de restaurar o seu conteúdo original. O ato de transformar uma mensagem em uma mensagem cifrada é chamado de ***encriptação*** e o ato contrário é chamado de ***decriptação***. Um método bastante simples de encriptação é o método ***Twist***, que requer que ambos remetente e receptor combinem uma chave secreta k, que deve ser um inteiro positivo.

O método Twist utiliza quatro vetores: *plaintext*, *ciphertext*, *plaincode* e *ciphecode*; sendo *plaintext* e *ciphertext* vetores de caracteres e *plaincode* e *ciphercode* vetores de inteiros. Todos os vetores são de tamanho n, onde n é o tamanho da mensagem a ser encriptada. Os vetores são iniciados na posição zero, de forma que os seus elementos são numerados de 0 a n-1. Para este problema, as mensagens apenas conterão letras maiúsculas, pontos e *underscores* (representando espaços).

A mensagem a ser encriptada é armazenada no vetor *plaintext*. Dada a chave k, a encriptação é feita da seguinte forma. Primeiro, as letras em *plaintext*

são convertidas em códigos inteiros que são armazenados em *plaincode*. A conversão é feita da seguinte forma: '_' = 0, 'a' = 1, 'b' = 2, ..., 'z' = 26 e '.' = 27. Depois, cada código em *plaincode* é convertido em um código encriptado em *ciphercode* através da seguinte fórmula: para todo i de 0 a n-1,

ciphercode[i] = (*plaincode*[(k*i) mod n] - i) mod 28.

(Aqui x mod y é o resto positivo da divisão de x por y. Por exemplo, 3 mod 7 = 3, 22 mod 8 = 6 e -1 mod 28 = 27. Você pode usar o operador '%' do C, desde que você some y caso o resultado seja negativo.) Por último, os códigos obtidos em *ciphercode* são convertidos novamente em letras (pela mesma regra descrita anteriormente) e são armazenadas em *ciphertext*.

A encriptação pelo método Twist da mensagem 'cat' com chave 5 é ilustrada na tabela a seguir:

Vetor	[0]	[1]	[2]
plaintext	'c'	'a'	't'
plaincode	3	1	20
ciphercode	3	19	27
ciphertext	'c'	's'	'.'

2. Tarefa

A sua tarefa é escrever um programa que decripta uma mensagem encriptada pelo método Twist. Por exemplo, dado o texto cifrado 'cs.' e um dicionário de palavras em português, o seu programa deve dar como resposta o texto original 'cat'. Note que você terá que descobrir "de alguma forma" a chave k usada para encriptar o texto. A chave k será um inteiro positivo menor ou igual a 300 tal que mdc(k, n) = 1.

OBS.: apesar dos exemplos serem em inglês, neste EP serão consideradas somente palavras contidas no dicionário (em português) fornecido no PACA.

3. Entrada de Dados

O arquivo DECRYPT.IN conterá um texto.

3. Saída de Dados

Um arquivo DECRYPT.OUT contendo a mensagem decriptada.

Exemplos

k TEXTO
11 cbmowxbkg => cachorro.
5 edrykonnklupposembjnugujqwki => este_e_um_teste_muito_chato.
29 edcnkjyzmjrpianvbyzo => espero_que_funcione.

Observações:

- Somente comandos ensinados em aula poderão ser usados no EP.
- O cabeçalho do EP 3 deve ser:

```
/* ************************************************************************ */
/* Nome: [coloque aqui seu nome]                                            */
/* Numero USP: [coloque aqui seu numero USP]                                */
/* Exercício-programa 3                                                     */
/* ************************************************************************ */
```

- EPs que não compilam receberão nota ZERO. O comando a ser usado na compilação do monitor será:
  gcc –Wall –ansi –pedantic –O2 –o ep3 ep3.c
  Certifique-se que seu EP compila no sistema operacional Linux com o comando acima. Mensagens de "warnings" serão penalizados na nota.
- Não serão aceitos EPs atrasados. Será considerado como EP não entregue.
- Você deve entregar somente o arquivo contendo o código fonte: *.c
  Outros arquivos que não sejam .c entregues "por engano" receberão nota ZERO.
- Seu programa NÃO precisa checar consistência de dados.
- O EP deve ser feito de forma INDIVIDUAL. Você pode conversar e discutir a solução com seus colegas, mas em hipótese alguma você deve mostrar e/ou ver o código dos outros. Qualquer problema com o código do EP deve ser tratado com o monitor da disciplina.
- EPs copiados parcialmente ou totalmente da internet ou de qualquer outra fonte será considerado plágio. EPs que tentem "mascarar" a cópia também serão considerados plágio.
- EPs com plágio receberão nota ZERO, o aluno será REPROVADO e seu nome será encaminhado a Comissão de Graduação.