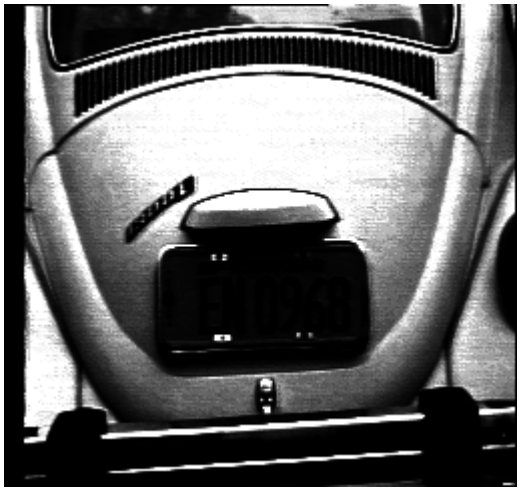
Trabalho 4 – ICC 1 (SCC-221) - 2017

CSI – A missão

Finalmente, após inúmeras tentativas frustradas, Dick Vigarista pôde finalmente comemorar a sua primeira vitória à frente de Peter Perfeito, Irmãos Rocha, Professor Aéreo, Rufus Lenhador, Penélope Charmosa, Quadrilha da Morte, dentre outros intrépidos corredores.

Aconteceu, porém, que um vigarista infinitamente pior, Joesley Safadão, resolveu denunciar o nosso astro-vilão da "Corrida Maluca" por excesso de velocidade! Sim. A corrida vencida por Dick Vigarista tinha por regra não ultrapassar a velocidade máxima de 180 km/h. Joesley Safadão, com livre trânsito nos mais altos escalões do governo, subornou o chefe do serviço de trânsito local (Michel Tâmara) e levou dos arquivos a imagem abaixo, que caiu como uma bomba entre os competidores. A imagem foi capturada com o auxílio de um radar.



Acontece que a câmera é de quinta categoria e a placa do carro não pode ser vista a olho nu. Entretanto, a imagem revela claramente uma traseira de fusca (mesmo modelo e cor do carro de Dick Vigarista !) e o radar que disparou a câmera, indica a incrível velocidade de 295 km/h. (Sim… Dick tinha sob a corroceria de seu fusquinha 66, um motor V8 turbinado !!!!)

Dick nega veemente qualquer malfeito. Ele garante que naquele momento rezava um terço junto com os seus fiéis amigos Renan, Aécio, Dirceu, Lula, Eduardo, Maluf, JBF e Odebrecht.

Em meio a toda celeuma, a Polícia Federal (PF) contratou você, para desvendar este mistério: se a placa do automóvel for revelada, Dick pode, mais uma vez, ver seu sonho de vencer uma corrida ir por água abaixo. E Penelope Charmosa, a segunda colocada, levantará o troféu de vencedora!

Só que a PF quer que você faça um programa um pouco mais amplo, capaz de ser usado também em outras situações. A entrada fornecida pela PF tem o seguinte formato:

Operação
NomeArquivo

Operação: é o tipo da função que você deve aplicar sobre a imagem e que os técnicos da PF acreditam será capaz de revelar informações escondidas (neste caso a placa do fusca).

Operação == 1 → operação de log. (ver detalhes mais adiante)
Operação == 2 → operação de "esticar" o constraste. (ver detalhes mais adiante)
Operação == 3 → "baldinho do paint brush" (ver detalhes mais adiante)

NomeArquivo é nome da imagem no formato <*.pgm> onde '*' é qualquer cadeia de caracteres (sem espaços).

O formato de imagem PGM (http://netpbm.sourceforge.net/doc/pgm.html) é um dos mais simples que existem e armazena imagens em nível de cinza apenas, com pixels de profundidade 1, isto é, cada pixel tem exatamente um BYTE podendo ter valores entre 0 – 255 (em C, qual o tipo mais "econômico" possível para representá-los??). O cabeçalho da imagem (mais conhecido como ***header***) tem o seguinte formato:

P<**n**>
Comentario qualquer: normalmente o software que criou a imagem !
XDIM YDIM
MaxVal
<os pixels da imagem de fato>

Onde **n** pode ser == 2: imagem é armazenada textualmente, ou seja, no formato ASCII.
== 5: imagem é armazenada no formato binário.

 # qualquer coisa……. É uma linha de comentário, indicando o "fabricante" da imagem !

XDIM é um inteiro que representa a quantidade de colunas da imagem
YDIM é um inteiro que representa a quantidade de linhas na imagem.

XDIM e YDIM estão separados por um espaço em branco.

MaxVal é o valor do maior pixel encontrado na imagem (normalmente 255).

Assuma que a imagem é representada por uma matriz com as seguintes convenções:

Eixo Y (YDIM = 7)

(0,0)	(1,0)								(9,0)
(0,1)									
(0,2)									
									(x,y)
(0,7)									(9,6)

Eixo X : (XDIM = 10)

Todos estas linhas e comentários têm no máximo 99 caracteres.

Finalmente temos os pixels da imagem: <os pixels da imagem de fato>

Caso a imagem seja do tipo P2 (ASCII), na linha abaixo de **MaxVal** teremos os pixels da imagem (valores inteiros entre 0 e 255) UM EM CADA LINHA.

Caso a imagem seja P5 (binária), na linha abaixo de **MaxVal** teremos um STREAM de BYTES de tamanho XDIM * YDIM, que contém todos os pixels da imagem.

Veja exemplos de cada uma destes formatos em:

Fusca no formato ASCII (http://www.lcad.icmc.usp.br/~jbatista/scc221/fusca.pgm)
Fusca no formato Binário (http://www.lcad.icmc.usp.br/~jbatista/scc221/fuscaBIN.pgm)

Faça o download de ambas as imagens e visualize no viewer de sua preferência. São exatamente iguais, apenas armazenadas de formas diferentes! Agora abra-as em um editor de texto. Tente entender por que a imagem binária aparece como "lixo" no editor! Qual a explicação ????

A Saída

Seu programa deve gerar como saída uma imagem processada no formato PGM, SEMPRE NO FORMATO ASCII (mesmo que a imagem original seja no formato binário).

O comentário na segunda linha do cabeçalho da imagem que você deverá gerar deverá ter exatamente o seguinte conteúdo:

"# CREATOR: Image Generator SCC-221 - ICC I" (sem as aspas)

Sobre os requisitos do trabalho:

a) Utilize uma **struct** para organizar o seu tipo "imagem": esta **struct** deve conter: as dimensões X e Y da imagem; os valores do pixel de maior valor e o de menor valor da imagem (estes serão úteis para realizar as operações de *log* e *stretching*); o tipo da imagem (binário ou ASCII) e claro, os pixels de sua imagem – QUE DEVER SER ALOCADA DINAMICAMENTE! Você pode usar tanto uma estrutura bi-dimensional quanto unidimensional.

b) utilize funções para tudo: operações tais como loadimage, saveImage, calcula min e max, libera imagem, e as operações solicitadas devem ser funções !!! Tente manter o seu código em main() o mais curto possível, sempre chamando funções para realizar as operações necessárias. Mantenha o número de argumentos reduzido, passando a sua struct como um dos argumentos.

c) evite variáveis globais. Não há necessidade de usar NENHUMA variável global.

Sobre as operações:

consulte o link http://homepages.inf.ed.ac.uk/rbf/HIPR2/

e selecione: worksheets →Point Operations. Lá você encontrará links para (Contrast stretching e Logarithm Operation). São operações MUITO simples de entender e implementar. Explicaremos em sala de aula também.

Log: deve obedecer a seguinte equação **y = c * log(1+y)** com **c = 255 / log(1+max),** onde:
y é todo pixel da nova imagem; x é todo pixel da imagem original e max é o pixel de maior valor da imagem original.

Contrast Stretching (Expansão de Contraste): y = (x-min) / (max-mim) * 255, onde:
max e mim são os maiores e menores pixels da imagem original x.

A operação abaixo não é da mesma categoria das duas descritas acima.

Baldinnho do painbrush: formalmente denominada de "**flood fill**". Veja explicação em: https://en.wikipedia.org/wiki/Flood_fill

Informaçãoes importantes para a implementação da operação flood fill:

a) Você deverá implementar uma **versão recursiva**.
b) o pixel onde o baldinho inicialmente começa a pintar é exatamente o pixel central da imagem, facilmente calculado a partir das dimensões X e Y da mesma!
c) A região a ser pintada está delimitada por um contorno fechado, cujo valor do pixel é igual a 255. Veja a imagem http://www.lcad.icmc.usp.br/~jbatista/scc221/folha2.pgm
d) a "cor" com a qual o interior deste contorno deverá ser pintado é o nivel de cinza = 100.
e) você deve "expandir" a pintura pintando somente os pixels acima, abaixo, à esquerda e à direita do pivel corrente.

Há 5 casos de testes neste trabalho. Cada um utiliza uma imagem distinta. As imagens listadas abaixo correspondem aos casos 1, 2, 3, 4 e 5, respectivamente.

http://www.lcad.icmc.usp.br/~jbatista/scc221/fusca.pgm
http://www.lcad.icmc.usp.br/~jbatista/scc221/station.pgm
http://www.lcad.icmc.usp.br/~jbatista/scc221/sat_image.pgm
http://www.lcad.icmc.usp.br/~jbatista/scc221/bubbles2.pgm
http://www.lcad.icmc.usp.br/~jbatista/scc221/folha2.pgm

Boa divertimento!
E contribua com o Brasil, expondo os vigaristas que mantém esta nação no atraso.