Lista de Exercícios 1

1. Considerando a imagem 6x6 abaixo, faça:

100	200	50	20	222	44
15	14	13	12	19	22
60	3	50	0	9	8
7	6	5	4	3	2
1	0	22	33	44	55
66	77	88	99	101	102

   a) Crie um vetor V para represar os valores dos pixels da imagem.
   b) Qual o valores das posições V[12] e V[20]?

2. Crie uma imagem 4x4 com base no vetor V[16] abaixo:

V = | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 | 0 | 1 | 2 | 3 | 4 | 5 |

3. Dado o vetor V[20] que armazena os valores dos pixels de uma imagem 5x4, informe os valores dos pixels nas coordenadas (3, 2) e (2,1)

V = | 9 | 8 | 1 | 2 | 3 | 6 | 4 | 50 | 90 | 70 | 10 | 30 | 40 | 60 | 20 | 80 | 17 | 87 | 62 | 91 |

4. Crie uma função para criar uma imagem. A função deve conter os seguintes parâmetros:

```
void criar(PGM *pgm, int largura, int altura, unsigned char corFundo);
```

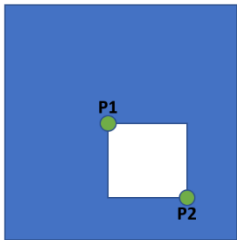
5. Crie uma função para preencher uma determinada linha de uma imagem PGM com uma cor específica (branco, cinza ou preto).

```
void setLinha(PGM *pgm, int linha, unsigned char cor);
```

6. Crie uma função para verificar se as coordenadas de um pixel passadas por parâmetro são válidas.

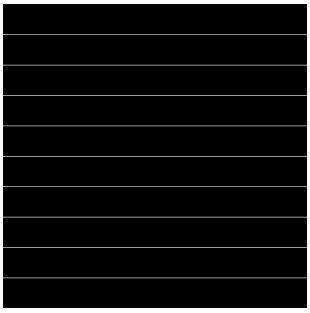
```
bool coordValida(PGM *pgm, int x, int y);
```

7. Crie uma função para preencher uma região de uma imagem PGM com uma determinada cor (branco, cinza ou preto). A região é definida pelas coordenadas $P_1(x_1, y_1)$ e $P_2(x_2,y_2)$.



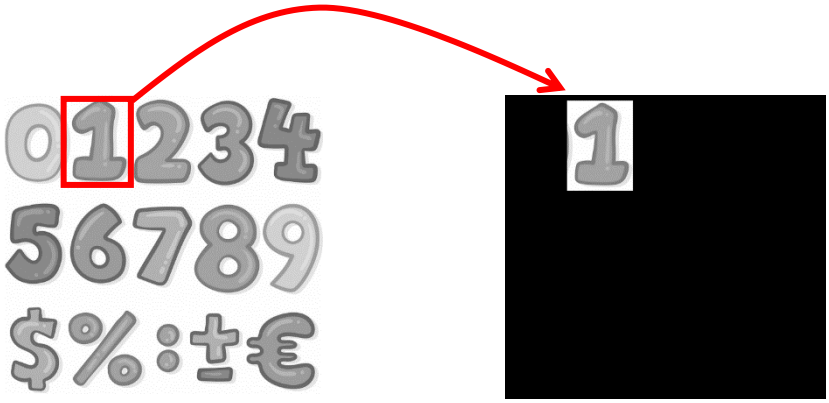
Observação: preencher a região somente das coordenadas válidas (utilize a função criada no exercício 6).

8. Crie uma imagem PGM de 500x500 pixels com fundo preto (colocar o valor zero para todos os pixels). Em seguida, defina uma linha branca a cada 50 linhas da imagem.



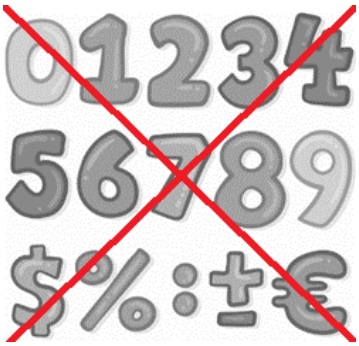
9. Crie um método para inverter a imagem verticalmente (*flip*).

10. Considere uma imagem PGM de entrada e uma região definida pelas coordenadas $P_1(x_1, y_1)$ e $P_2(x_2,y_2)$. Desenvolva uma função para criar uma imagem de saída, a qual possui a mesma dimensão da imagem de entrada e a cor de todos os pixels é preta. Em seguida, copie a região definida na imagem de entrada para a imagem de saída.



11. Desenvolva uma função em C++ para desenhar uma borda em uma imagem PGM. A cor e a espessura da borda devem ser passadas por parâmetro da função. Observação: A espessura da borda não pode ultrapassar 10 pixels.

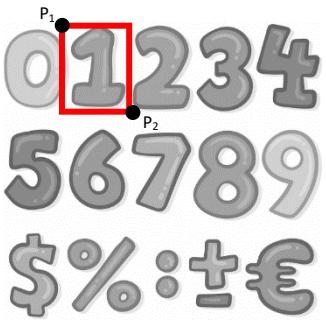
12. Desenvolva uma função em C++ para desenhar um “X” imagem PGM. A cor da linha deve ser passada por parâmetro.



13. Crie um programa em C++ para combinar duas imagens PGM de entrada I_1 e I_2 em uma única imagem PGM de saída I_s. Para os pixels $P_1(x, y)=C_1$ da imagem I_1 e $P_2(x, y)=C_2$ da imagem I_2, ambos com as mesmas coordenas x e y, as cores devem ser combinadas como segue:
    - Média: $(C_1+C_2) / 2$
    - Maior: $\max(C_1,C_2)$
    - Menor: $\min(C_1,C_2)$

    Observações: Considere que todas as imagens possuem a mesma dimensão; Crie um menu que permita o usuário escolher a combinação desejada.

14. Considere uma imagem PGM de entrada e implemente uma função em C++ para criar uma borda em uma região definia por dois pontos, P_1 e P_2. A função deve receber como parâmetro as coordenadas dos pontos e a cor da borda.



15. Crie um programa em C++ para converter os tons de cinza de uma imagem PGM de entrada em uma imagem preto e branco de saída:
    - Se a cor do pixel na imagem de entrada for maior que 128, sua cor será 255 na imagem de saída;
    - Se a cor do pixel na imagem de entrada for menor ou igual a 128, sua cor será 0 na imagem de saída.

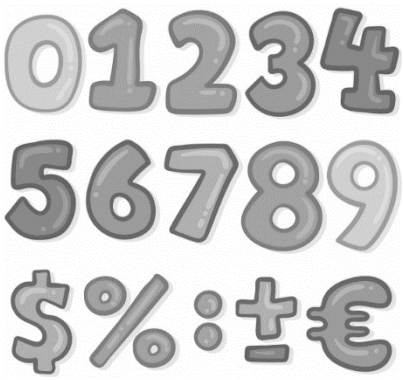


Imagem de Entrada

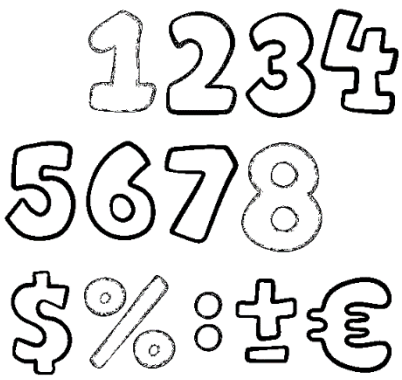


Imagem de saída