Os critérios de correcção são os seguintes:

- Grupo I
  - Erros (7 erros):
    - O tipo de retorno da função é int e não float
    - No primeiro condicional, o operador de igualdade == (correto) foi trocado pelo da atribuição = (incorreto)
    - No primeiro condicional existe um ponto e vírgula após o condicional, o que significa instrução vazia, distinto do que o programador pretende, ter a instrução return associado ao condicional.
    - No primeiro condicional e instrução associada, a variável x está trocada com a variável y
    - Na instrução de return final, o operador do resto da divisão % (correto) foi trocado pelo da divisão inteira (incorreto)
    - Na instrução de return final, na última expressão tem as variáveis x e y trocadas.
    - Na instrução de return final, falta um ponto e vírgula.
  - Cotação:
    - 0,5 por cada erro detectado, até ao máximo de 3 valores. Notar que há 7 erros, pelo que perdoa-se a falha de um erro.
    - -0,5 por erro introduzido não existente (cancela um erro bem identificado)
    - 3 valores para quem corrige o valor de retorno para int, e faz return 1; (considerando que é válida a gralha existente, que se pretende calcular o mínimo divisor comum)
  - A substituição do código por outro, com a funcionalidade implementada, não responde ao solicitado, que é a identificação dos erros no código fornecido. Nestes casos a pergunta poderá ser valorizada até metade, e no caso de serem apontados erros ao código original, é valorizada essa parte da resposta.
  - É indiferente a colocação do else no if.
  - Houve quem tivesse colocado o código corrigido, e quem indique os erros. Ambas as formas são válidas.
- Grupo II
  - Critérios positivos (soma de 0 a 1,5) - apenas se o exercício estiver muito mal
    - + 0,5 - assinatura correta
    - + 0,5 - retorno da função correto
    - + 0,5 - lógica relativamente correta
  - Critérios negativos (subtrai de 3 a 1,5) - quando o exercício está globalmente bem
    - - 0,5 - erro no código (expressão numérica/lógica incorreta; instrução incorreta)
    - 0 - um erro sintático (inclui troca do operador == pelo =)
    - - 0,5 - dois ou mais erros sintáticos menores.
    - -1 - permite gerar números repetidos (no caso de teste de repetidos parcial, só com o anterior, por exemplo, a penalização fica em 0,5)
  - Este exercício é muito parecido com a AF Baralhar. A versão mais simples, para quem não se lembre desta AF (utilizada também nos e-fólios), seria ir gerando valores no intervalo solicitado, e verificar se o valor já foi gerado. Não houve provas recebidas com a versão parecida com a atividade formativa, que seria também a mais

eficiente (complexidade linear contra complexidade quadrática da solução adoptada).
    - Vários estudantes a retornarem o vetor. A função é chamada como sendo um procedimento, não retorna nada, apenas inicializa o vetor que recebe no primeiro argumento.
    - Vários estudantes a colocarem um vetor de dimensão 5 no primeiro argumento, quando a função tem de receber um vetor de dimensão N, pelo que não deveriam indicar qualquer dimensão.
    - Alguns estudantes a solicitarem ao utilizador na função, os valores recebidos nos argumentos. Se são recebidos, já os têm pelo que não faz sentido pedir o que já se tem. Há aqui algum tipo de confusão que era importante identificar e esclarecer.
- Grupo III
    - Critérios positivos/negativos idêntico ao grupo anterior.
    - A estrutura de dados espectável é uma matriz de 5x5, sendo ignorada a posição central.
    - A função para mostrar a carta, apenas tem de ter um condicional e colocar uma marca na posição central.
    - A função para inicializar a carta, basta chamar em cada coluna a função do grupo anterior, podendo até ignorar a casa central, e utilizar uma expressão para definir os limites (não é necessário chamar a função 5 vezes, apenas uma vez).
    - Do programa esperava apenas a inicialização da semente aleatória, inicializar a carta, e mostrar a carta.
    - Troca de linhas por colunas, não penalizado.
    - Não penalizado soluções com uma variável para cada linha (ou coluna), resultando em código repetido, uma vez que é um erro de qualidade e não de funcionalidade.
    - Parte não realizada (uma das funções): 1 valor. No caso de faltar apenas o main, esta penalização não é aplicada (será apenas 0,5), apenas se faltar o procedimento para mostrar a carta ou inicializar a carta.
    - A colocação do typedef/struct incorreta, é considerada erro de sintaxe, sendo penalizado apenas se acompanhado de outro erro de sintaxe.
    - A não reutilização da função anterior, e possibilidade de geração de números repetidos, é considerado erro normal (0,5 de penalização).
    - Não considerar a casa central é considerado erro normal (0,5 de penalização)
    - A ordem das funções não é naturalmente levada em conta na resolução manual, não tem problema chamarem uma função definida mais à frente.
- Grupo IV
    - Critérios positivos/negativos idêntico ao grupo anterior.
    - A solução mais esperada, também a mais simples, seria de zerar os valores saídos, seguida de teste às linhas, colunas e diagonais, para ver se todos os valores estão zerados.
    - Os números saídos são de tamanho arbitrário (podem ser por exemplo 20 números). Penalização de 0,5 para quem considerou este valor fixo.
    - Os números enviados podem também estar por qualquer ordem. No caso do código assumir a ordem da carta, penalização de 1 valor.
- Grupo V (2 valores por cada uma das 4 alíneas)

- 0,5 - por não permitir uma pessoa ter envolvimentos distintos em ocorrências distintas
- 1 - por não ler/gravar informação para ficheiro. Valorizados os métodos criados de manipulação da estrutura de dados.
- 1 - por fazer relatório com apenas os totais, e não informação diária
- Esperava-se um tipo Delito (com ano, mês, dia, montante, envolvimentos); um tipo Envolvimento (com pessoa, tipo de envolvimento), e os tipos opcionais: envolvimento (nome) e pessoa (nome). O mais simples seria apenas dois tipos, para facilidade, embora o correto seria começar logo com os 4 tipos. Todos os tipos na forma de lista, com apontador para o seguinte.
- Na gravação para ficheiro, fazer uma função para cada tipo, em que o último elemento coloca uma marca de fim de lista. As funções inversas para leitura de ficheiro, juntamente com um construtor do tipo, ou seja, uma função para criação de cada tipo de elemento, e outra função para libertar toda a lista.
- No relatório mensal, com a lista de delitos, contar (ou copiar, depende das opções anteriores) para um vetor com o número dos dias, as ocorrências que dizem respeito a cada dia do mês selecionado, por tipo de envolvimento (uma lista por cada dia do mês). Após a contagem, produzir o relatório e libertar a estrutura auxiliar.
- Fazer uma estrutura auxiliar com um montante por pessoa. No final, ordenar a lista por montante, e mostrar os 10 primeiros elementos.

Dei algumas indicações sobre a resolução, e tendo dúvidas sobre a nota, convém solicitar a cópia para verem concretamente onde foram aplicados os critérios de correção. Se detectarem algum aplicado incorretamente, devem solicitar revisão de prova. Os critérios utilizados são os que estão na mensagem de lançamento de notas.

Fica aqui resoluções possíveis a adicionar às indicações já fornecidas, também no mesmo estilo que as indicações de resolução dos e-fólios, não divulgando um ficheiro que compilem e executem. Se tiverem questões coloquem:

Grupo I era a atividade formativa mdc.c, basta que abram as AFs resolvidas e vêm lá o código:

```
int mdc(int x, int y)
{
    /* caso y=0, retornar x */
    if(y==0)
        return x;
    /* c.c. retornar mdc(y,mod(x,y)), exactamente o comando seguinte
        após substituir o mod pelo comando do resto da divisão % */
    return mdc(y,x%y);
}
```

Os 7 erros que existiam no enunciado já estão indicado sem cima.

Neste grupo a média de notas foi de 2,2 valores em 3 possíveis, tendo sido o grupo com maior média.

No grupo II pretendia que fizessem algo parecido com o baralhar.c, algo deste tipo:

```
void GeraVetor(int *sequencia, int N, int A, int B)
{
        // versão mais parecida com a atividade formativa Baralhar
        int i, j, auxiliar[100];

        // inicializar o vetor auxiliar com os números possíveis

        for (i = 0; i < B - A + 1; i++)

                auxiliar[i] = A + i;

        for (i = 0; i < N; i++) {

                j = i + rand() % (B - A + 1 - i);

                sequencia[i] = auxiliar[j];

                auxiliar[j] = auxiliar[i];

        }

}
```

Reparem que ao utilizar o vetor auxiliar, não é necessário fazer a troca de valores como na AF baralhar, e estaria mais correto alocar o vetor com a dimensão exacta (A-B+1), mas em prova podem fixar um valor máximo sem problema. Mas a maior parte (ou julgo mesmo que todos), resolveram da seguinte forma:

```
void GeraVetor(int *sequencia, int N, int A, int B)
{
```

```
        // versão simples mas com dois ciclos aninhados
        int i, j;
        for (i = 0; i < N; i++) {
                sequencia[i] = A + rand() % (B - A + 1);
                for (j = 0; j < i; j++)
                        if (sequencia[j] == sequencia[i])
                                break;
                if (j < i)
                        i--;
        }
}
```

Este grupo acabou por ter uma média de 1,76 em 3 valores possíveis, esperava uma média maior aqui. Embora seja um exercício colado a uma atividade formativa que foi útil também para um e-fólio, não foi mesmo assim fácil.

No grupo III eram solicitadas várias pequenas funções:

```
typedef struct {
        int numeros[5][5];
} BINGO;
void Inicializar(BINGO *carta)
{
        int i;
        for(i=0;i<5;i++)
                GeraVetor(carta->numeros[i], 5, 1+15*i, 15+i*15);
        // ignorar o número no meio, ou colocá-lo a zero
        carta->numeros[2][2] = 0;
}
void MostraBingo(BINGO carta)
{
        int i, j;
        for (j = 0; j < 5; j++) {
                printf("\n");
                for (i = 0; i < 5; i++)
                        if (j == 2 && i == 2)
                                printf("  X");
                        else
                                printf(" %2d", carta.numeros[i][j]);
        }
}
int main()
{
        BINGO carta;
        srand(1);
        Inicializar(&carta);
        MostraBingo(carta);
}
```

Pretendia que definissem o tipo, typedef, mas quem utilizou a matriz sem definir o tipo, serve também. A média de notas foi neste caso 1,86 valores em 3 possíveis. Este grupo teve uma média mais alta que o grupo anterior, o que se compreende já que as funções solicitadas eram muito pequenas.

O grupo IV houve poucas respostas corretas. Pretendia uma função deste tipo:

```
int TesteBingo(BINGO carta, int *numeros, int N)
{
	int i,j,k;
	// marcar a zero os números saídos
	for (i = 0; i < N; i++)
		for (j = 0; j < 5; j++)
			for (k = 0; k < 5; k++)
				if (carta.numeros[j][k] == numeros[i])
					carta.numeros[j][k] = 0;
	// verificar se existe uma linha toda a zero
	for (i = 0; i < N; i++) {
		for (j = 0; j < 5; j++)
			if (carta.numeros[i][j] != 0)
				break;
		if (j == 5)
			return 1;
	}
	// verificar se existe uma coluna toda a zero
	for (i = 0; i < N; i++) {
		for (j = 0; j < 5; j++)
			if (carta.numeros[j][i] != 0)
				break;
		if (j == 5)
			return 1;
	}
	// verificar se existe uma diagonal
	for (i = 0; i < 5; i++)
		if (carta.numeros[i][i] != 0)
			break;
	if (i == 5)
		return 1;
	// diagonal invertida
	for (i = 0; i < 5; i++)
		if (carta.numeros[i][4-i] != 0)
			break;
	if (i == 5)
		return 1;

	return 0;
}
```

Julgo que esta seria a forma mais direta, a realizar em prova. Marcar o que se pretende detectar, e depois validar. Houve quem tivesse somado as linhas/colunas/diagonais, e

verifique no final se era zero. Essa forma é mais elegante que esta, já que deixa o teste para o final. Houve também versões mais complexas. A média final nesta pergunta foi de 1,02 valores em 3 possíveis, mas contabilizando como zero os estudantes que não realizaram esta pergunta.