Os alunos são responsáveis pela entrega da resolução, dentro do prazo estipulado. Exercícios entregues fora de hora são considerados desistências.

A hora do UMAIA é a hora padrão.

Enunciado e pré-requisitos gerais

Leia atentamente **todo** o enunciado antes de começar a responder à prova.

- Nas componentes práticas, utilize apenas, estruturas de dados simples e constituídas por si, sem recorrer a objetos do .NET, como *List*, *ArrayList*, etc.
- Recorra apenas a *Console app (.NET framework)*.
- Codifique com o *Visual Studio* instalado no laboratório.

Grupo 1: dry-running (5 valores)

Assine e apresente, no verso desta folha, um *dry-running* correspondente à execução do algoritmo do programa abaixo, o qual faz a procura de uma amostra de caracteres num texto. Considere os seguintes valores nas variáveis, ou argumentos:

- **amostra**= "sus"
- **texto**= "Ética versus moral"

```
static int Procura(string amostra, string texto) {
    int j, M = amostra.Length;
    int i, N = texto.Length;
    for (i = 0, j = 0; i < N && j < M; i++) {
        if (texto[i] == amostra[j])
            j++;
        else {
            i -= j; j = 0;
        }
    }
    if (j == M)
        return i - M;
    else
        return N;
}
```

Algorithms 4th edition. Sedgewick, Robert, et all

Requisitos obrigatórios:

1) Desligue ou coloque em modo silencioso o seu telemóvel.
2) Faça *login* utilizando as credenciais indicadas pelo docente.
3) Execute o comando de obtenção da prova, indicado pelo seu docente. A execução desse comando irá constituir uma pasta **c:\A0xxxxx** com o seu número mecanográfico.
4) Coloque nessa pasta os ficheiros de resposta aos grupos de exercícios acima, identificando o ficheiro pelo número do grupo e número da questão. A prova é com consulta, **donde se penaliza em 50% caso exista algum erro de sintaxe** (ou erros de compilação), no código produzido.
5) Deverá guardar todos os exercícios na pasta indicada.
6) No fim execute o comando de submissão da prova, indicado pelo seu docente.
7) Cada aluno é responsável pela entrega da sua prova, a qual deverá ser entregue antes do fim do tempo regulamentar.

Número	Assinatura / nome
A0	

Grupo 2: algoritmia simples (5 valores)

A fórmula de Lorentz resulta do estudo do físico e matemático Lorentz para calcular o peso ideal de uma pessoa em quilogramas, em função da sua altura, expressa em centímetros.

$$Peso = (altura - 100) - \frac{altura - 150}{K}$$

Onde "Peso" representa o peso ideal; K deve tomar os valores: 4, se for homem, e 2 se for mulher.

Defina de raiz a sua solução e coloque-a, desde logo, na pasta de resolução da prova. Faça uma aplicação em consola, (.Net *framework*), para ler os necessários valores de entrada dos argumentos para **o método** que implementa a função de Lorentz. No fim apresente o resultado, com o peso ideal.

Grupo 3: algoritmos sobre estruturas de dados em OO (10 valores)

Defina de raiz a sua solução e coloque-a, desde logo, na pasta de resolução da prova. Pretende-se que escreva a solução (tipo *console*), tendo em conta que deve desde logo ajustar o *namespace* para "`ResolucaoG3_a0xxxxx`", onde xxxxx corresponde ao seu número mecanográfico (**0.5 valor**).

Deseja-se que implemente uma solução C# (consola) para representar uma agenda de contactos de amigos, através da noção de conjunto, implementado por si, e com base num *array* (vetor de 5 posições). Cada contacto (amigo) deverá ser colocado na próxima posição livre do *array,* até que este fique cheio. Ao ficar cheio deve crescer em 5 novas unidades, aceitar novos contactos, e assim sucessivamente.

A sua agenda (ou sistema), deverá ser capaz de implementar as seguintes funcionalidades, respeitando na íntegra a seguinte assinatura de métodos:

`void InserirContacto(`
`  string nome,`
`  long telefone)`
	Para permitir o registo de um novo contacto de um seu amigo. Deverá validar se não se trata de um contacto já presente; cada contacto deve conter o nome completo, a data de nascimento, e o número do telemóvel (chave) **(3 valores)**.

`void AtualizaNumero(`
`  long numAtual,`
`  long numNovo)`
	Para permitir a substituição de um número de telemóvel de um contacto registado **(3 valores)**.

`void EliminarContacto(` `  long telefone)`	Elimina o contacto, dado o seu número de telefone; internamente deve compactar o *array* que representa o conjunto dos contactos de amigos **(3 valores)**.
`override string` `ToString()`	Deve devolver uma *string* com toda a informação presente na agenda (o conjunto dos contactos) **(1.5 valores)**.

A qualidade do código aos grupos/alíneas anteriores, mesmo que funcione, estão sujeitas à análise e respetiva valoração, de acordo com a sua qualidade, perfeição e objetividade.

Boa sorte!