Nome:	Número:	Assinatura:

Universidade do Algarve

Programação Imperativa

Segunda Festa de Programação Imperativa

`http://deei-mooshak.ualg.pt/~pi`

Aviso Geral

- Use o computador **exclusivamente** para escrever, compilar e submeter programas `C` ao `mooshak`. **Tolera-se** a consulta dos acetatos das aulas teóricas, os ficheiros apresentados nas aulas e as resoluções aos exercícios das fichas práticas arquivadas no disco. Qualquer outro uso do computador que não seja resolver os problemas propostos, **não é autorizado** durante a festa e **qualifica-se como fraude**.
- Qualquer uso de material indevido (telemóveis, *chats*, pdfs etc...) é sancionado com **reprovação imediata à UC de Programação Imperativa e será assinalada às autoridades académicas competentes**.
- Qualquer comportamento indevido, não autorizado ou fraude académica, etc... **é sancionado com reprovação imediata à UC de Programação Imperativa e será assinalada às autoridades académicas competentes**.
- A elegância e eficiência do código serão elementos de avaliação. Por exemplo, recorra à definição de funções sempre que justificado.
- Um exercício pode exigir uma determinada solução (por exemplo, *"não usar o tipo float"*, *"não usar ciclos"*, etc.). Uma solução aceite pelo `mooshak` mas que não respeita estas exigências será avaliada **para metade da sua cotação**.

As regras do mooshak

- `gcc -Wall -lm ...` A função `main` **deverá sempre devolver 0**.
- Uma linha de input ou de output **termina sempre com** `\n`.
- input/output: Não há espaços a não ser os que estão explicitamente referidos no enunciado.
- **Deixe sempre uma linha em branco** no fim do ficheiro C submetido.

Exercício 1 (Problema A - 1 pontos)
O desafio neste exercício é escrever um programa `C` *que leia um número inteiro (**positivo ou negativo**) e que escreva como output o maior dos seus algarismos, **usando exclusivamente a recursividade**.*

input: *Uma linha com um inteiro n, compatível com o tipo* `int`*. Por exemplo:*

```
15231
```

output: *Uma linha com o algarismo pretendido.*

Por exemplo (em resposta ao exemplo de input anterior):

```
5
```

Outro exemplo:

com a entrada

```
-62841
```

a saída é:

```
8
```

□

Exercício 2 (Problema B - 1.25 pontos)

Considere a sequência de inteiros definida pela relação de recorrência seguinte:

$$P(n) = \begin{cases} 0 & se\ n = 0 \\ 1 & se\ n = 1 \\ 2 \times P(n-1) + P(n-2) & caso\ n > 1 \end{cases}$$

escreva um programa `C` *que define uma função* ***exclusivamente recursiva*** *(sem recurso aos ciclos) para encontrar o n-ésimo elemento desta sequência. Deverá usar o tipo* `long long int` *para os cálculos numéricos e a* ***recursividade terminal*** *para a eficiência dos cálculos.*

input: *Uma linha com um inteiro n. É garantido que se dá um inteiro positivo ou nulo. Por exemplo:*

```
8
```

output: *Uma linha com o n-ésimo elemento da sequência numérica. Não há casos anómalo por considerar aqui.*

Por exemplo (em resposta ao exemplo de input anterior):

```
408
```

□

Exercício 3 (Problema C - 1.75 pontos)

Há números que gostam de si próprios. Vejam só esses: um número natural n (ou seja $n \in \mathbb{N}$) tal que este, sendo d o número de dígitos de n, seja igual a soma dos seus dígitos potência d. É um número inteiro muito centrado nele próprio...

Por exemplo, 153 *é um destes números, porque tem* 3 *dígitos e* $153 = 1^3+5^3+3^3$. *Assim como* 1634 *porque tem comprimento* 4 *e* $1634 = 1^4 + 6^4 + 3^4 + 4^4$.

O objectivo este exercício é escrever um programa em C que inclua uma função ***recursiva*** *(ou seja,* ***sem ciclos****) que consiga determinar se um inteiro é ou não um número que gosta de si-mesmo.*

input: *Uma linha com um inteiro natural (de tipo* `int`*, mas* ***garantidamente positivo ou nulo****). Por exemplo:*

```
153
```

Embora o input seja do tipo `int` *neste exercício, aconselha-se o uso do tipo* `long long int` *para os cálculos intermédios, para evitar de forma definitiva potenciais arithmetic overflows.*

output: *Uma só linha com a palavra* `YES` *se o input é um número que gosta de si-mesmo, ou a palavra* `NO` *em qualquer outro caso.*

Por exemplo (em resposta ao exemplo de input anterior):

```
YES
```

□

Exercício 4 (Problema D - 2 pontos)

Observe o seguinte padrão textual ASCII, cuja forma depende do tamanho da linha horizontal central.

```
tamanho 1   tamanho 2   tamanho 3   tamanho 4   tamanho 5   tamanho 6

                                                  *           **
                             *         **        ***         ****
  *          **             ***       ****      *****       ******
                             *         **        ***         ****
                                                  *           **
```

*Dado o tamanho n do maior padrão, que **se espera sempre positivo**, escreva um programa `C`, que, coma definição de funções auxiliares **exclusivamente recursivas** imprima linha a linha o seguinte padrão:*

```
|*
|**
| *
|***
| *
| **
|****
| **
|  *
| ***
|*****
| ***
|  *
|  **
| ****
|******
| ****
|  **
|  *
| ***
|*****
| ***
|  *
| **
|****
| **
| *
|***
| *
|**
|*
```

Nomeadamente, os padrões de 1 até n para depois seguir até 1. Neste exemplo, $n = 6$. Para facilitar a vizualização do resultado neste enunciado, o caracter `|` *assinala aqui o início da linha. **Ele não é parte do output!***

Aconselha-se que estruture a sua solução em funções auxiliares, uma para a escrita de um padrão e outra para a escrita da sequência de padrões. Poderá também querer dividir cada uma destas funções em duas: uma para a parte do padrão (resp. sequência) crescente e uma para a parte do padrão (resp. sequência) descrescente.

input: *Uma linha com um inteiro n (de tipo* `int`*). Por exemplo:*

```
6
```

output: *o padrão textual de tamanho n. Ou uma linha com a palavra* `NO` *em caso de anomalia. Por exemplo (em resposta ao exemplo de input anterior): o padrão dado acima (a sequência de diamantes 1,2,3,4,5,6,5,4,3,2,1).*

Com uma entrada negativa ou nula, deve ser assinalada uma anomalia.

□