Paradigmas da Programação II
LESI + LMCC (1^o ano)

3º Trabalho Prático

Ano Lectivo de 01/02

O trabalho deve ser realizado em grupo,
e deve ser entregue acompanhado dum relatório, por submissão electrónica,
até 6ªfeira, 03 de Maio.

1 Objectivos e Organização

Este trabalho prático tem como principais **objectivos**:

- continuação da exploração do computador em ambiente linux, e de algumas ferramentas de apoio à programação
- codificação, na linguagem imperativa C, de estruturas de dados e algoritmos mais complexos concebidos e execução e teste e dos programas escritos, incluindo manuseamento de ficheiros e textos;
- documentação dos projectos de programação realizados usando o processador LaTeX.

Para o efeito, esta folha contém seis enunciados, dos quais deverá resolver um.
O **relatório** a elaborar, deve ser claro e <u>sucinto</u> e, além do respectivo enunciado e da descrição do problema, deverá conter exemplos de utilização e o código do programa. Como foi dito, o relatório terá de ser escrito em LaTeX sugerindo-se que adoptem o esqueleto proposto que pode ser descarregado a partir da página WWW principal da disciplina, no URL:

`http://www.di.uminho.pt/~prh/curpp201.html`

Não se esqueça de indicar na capa do relatório, e como título, o nome do problema que escolheu; no campo *autores* deve indicar o nome completo e o número de cada elemento activo do grupo.

2 Enunciados

2.1 Extractor de Linhas

Pretende-se que desenvolva um programa que extraia n linhas de um ficheiro de texto e as imprima para o *stdout*. Estas n linhas podem ser extraídas do início, do fim ou do meio do documento, de acordo com os seguintes exemplos:

- `extrai 10 ficheiro` — extrai a décima linha do ficheiro;
- `extrai 10-20 ficheiro` — extrai do ficheiro as linhas 10 a 20 (inclusivé, logo 11 linhas);

- `extrai 20- ficheiro` — extrai desde a linha 20 até ao fim do ficheiro;
- `extrai -10 ficheiro` — extrai as primeiras dez linhas do ficheiro;

No caso do ficheiro não ser indicado, o programa deve considerar que os dados serão lidos a partir do *stdin*.

2.2 Extractor de Colunas

Pretende-se que desenvolva um programa que extraia m colunas de n caracteres de um ficheiro de texto e as imprima para o *stdout* separadas pelo caracter *tab*. Estas colunas serão extraídas de acordo com os seguintes exemplos:

- `extrai -10 ficheiro` — extrai uma coluna com dez caracteres, a contar do início da linha;
- `extrai 10-20 ficheiro` — extrai uma coluna com onze caracteres, desde o décimo ao vigésimo caracter do ficheiro;
- `extrai 20- ficheiro` — extrai uma coluna, desde o vigésimo caracter até ao fim de cada uma das linhas.
- `extrai -10 20-30 ficheiro` — extrai duas colunas, uma do início até ao déciomo caracter e uma outra do vigésimo até ao trigésimo caracteres.

No caso do ficheiro não ser indicado, o programa deve considerar que os dados serão lidos a partir do *stdin*.

2.3 Pesquisador de Palavras

Deseja-se implementar um programa que dada uma palavra (ou expressão regular simplificada (pode usar-se o '`*`' para representar 0 ou mais caracteres nessa posição)) encontre todas as linhas que contém essa palavra, num ou mais ficheiros. O programa será invocado como se mostra a seguir:

```
> pesquisa 'sapo' ficheiro1 ficheiro2
```

Cada linha de resultado deve conter o nome do ficheiro onde foi encontrada a palavra, o número da linha onde ela se encontra, e a respectiva linha. Por exemplo:

```
> pesquisa 'sapo' ficheiro1 ficheiro2
ficheiro1:20:e ele comeu um sapo.
ficheiro1:30:o sapo morreu.
```

Como extra (que aconselhamos a implementar), considere as seguintes expressões regulares:

- `anan.s` — o '`.`' representa qualquer caracter (exactamente 1 caracter deve existir nessa posição). Desta forma, encontraria "ananás" correctamente escrito, ou a sua versão sem acentos, "ananas";
- `peras? verdes?` — o '`?`' representa que o caracter imediatamente antes pode ou não existir. Neste caso, quer o singular quer o plural seriam encontrados;
- `laranja$` — o '`$`' significa que não deve existir o que quer que seja após a palavra (a não ser o *new line*;

2.4 Formatador de Texto

Uma das tarefas mais incómodas ao escrever um texto, é o que se chama *line wrapping*, ou seja, quebrar linhas que excedem determinado comprimento em caracteres, tendo o cuidado de não quebrar palavras.

Deverá desenvolver um programa que dado um número de colunas $n \geq 20$ e um ficheiro (ou, dependendo do número de argumentos, o *stdin*) imprima para o *stdout* o texto depois de formatado.

Por exemplo, suponde que `ficheiro` contem o seguinte texto:

```
Este é um texto que deverá exceder as vinte colunas para que
sirva
de exemplo
```

então o comando

```
> extrai 20 ficheiro
```

deverá produzir um texto formatado como se ilustra abaixo:

```
Este é um texto que
deverá exceder as
vinte colunas para
que sirva de exemplo
```

2.5 Gráficos de Dispersão

Utilizando a biblioteca `gd`, o objectivo é desenvolver um programa, `desenha`, que leia do *stdin* um ficheiro de pontos (um ponto por linha, com as coordenadas (valores reais) separadas por um espaço) e represente os pontos graficamente. O programa deverá ajustar o tamanho da imagem e a escala de acordo com o que achar necessário.

A imagem gerada deve ser gravada num ficheiro cujo nome será passado ao programa como primeiro argumento. Uma utilização típica do programa será:

```
> cat pontos | desenha grafico.png
```

Para documentação (e possível *download*) da biblioteca `gd` pode consultar

`http://www.boutell.com/gd/`