**No final deverá confirmar que submeteu corretamente no SIGEX o código fonte dos seus programas utilizando o nome indicado no enunciado.**

**Quaisquer cópias detetadas serão penalizadas com anulação da prova.**

1. Tendo por base as bibliotecas de estruturas de dados apresentadas em Programação 2, implemente as funcionalidades pedidas nas duas alíneas seguintes no ficheiro **prob1.c**. Sempre que conveniente utilize as funções disponíveis nas estruturas lista ligada e fila.
   1. [15 pontos] Implemente a função vetor\_inverte que inverte a ordem dos elementos de um vetor.

void vetor\_inverte(vetor \*v)

O parâmetro é o apontador para o vetor a inverter e que deverá ser verificado.

Indique ainda num comentário no início do código da função qual a complexidade do algoritmo que implementou (não é necessário justificar).

Depois de implementada a função, o programa deverá apresentar:

Vetor Original: Um -> Dois -> Tres -> Quatro -> Cinco

Vetor Invertido: Cinco -> Quatro -> Tres -> Dois -> Um

* 1. [10 pontos] Implemente a função subfila\_maior que retorna uma fila nova com os elementos da fila original cuja *string* seja maior (em comprimento) do que um limiar.

fila\* subfila\_maior(fila \*f, int limiar)

O primeiro parâmetro é o apontador para a fila inicial, o segundo é o valor que define o limiar de comparação; a função deve retornar a nova fila ou NULL se não for bem sucedida. Depois de terminada a função, a fila inicial deve manter os elementos pela mesma ordem (isto é, se forem retirados, devem ser repostos).

Depois de implementada a função, o programa deverá apresentar:

Fila: Amarelo | Verde | Preto | Azul | Laranja | Vermelho | Branco

Sub-Fila maior que 6: Amarelo | Laranja | Vermelho

**\*\*\* Submeta o ficheiro prob1.c no SIGEX \*\*\***

1. Tendo por base as bibliotecas de estruturas de dados apresentadas em Programação 2, implemente as funcionalidades pedidas nas duas alíneas seguintes no ficheiro **prob2.c**. Sempre que conveniente utilize as funções disponíveis nas estruturas árvore AVL e grafo.
   1. [15 pontos] Implemente a função filtra\_strings para uma **árvore AVL** (definida pelo nó raiz) que imprime todas as *strings* cujo comprimento é maior do que um determinado valor.

int filtra\_strings(no\_avl \*no, int min)

O primeiro parâmetro da função é o apontador para o nó raiz da árvore e o segundo é o comprimento mínimo das *strings*. A função deverá também retornar o número de *strings* com comprimento maior do que min.

Depois de implementada a função, o programa deverá apresentar:

emirados arabes unidos

ilhas heard e mcdonald

ilhas virgens americanas

ilhas turcas e caicos

ilhas virgens britanicas

republica centro-africana

sao cristovao e neves

sao vicente e granadinas

* 1. [15 pontos] Implemente a função caminho\_pontos que, tendo por base um grafo, devolve o caminho mais curto que garante a passagem por um conjunto de pontos numa dada ordem.

int\* caminho\_pontos(grafo \*g, int \*pontos, int n, int \*tam\_caminho)

O primeiro parâmetro da função é o apontador para o grafo, o segundo é o vetor de pontos por onde o caminho deve passar (por ordem), o terceiro é o tamanho desse vetor, e o último parâmetro é usado para devolver o tamanho do caminho completo. Supondo que o vetor de pontos contém { 0, 3, 7 }, a função deverá retornar o caminho completo definido a partir de: caminho entre 0 e 3 e caminho entre 3 e 7 .

Os parâmetros de entrada devem ser verificados, e a função deve retornar NULL se não for bem sucedida.

Depois de implementada a função, o programa deverá apresentar:

Caminho completo: 0 4 3 2 1 5

**\*\*\* Submeta o ficheiro prob2.c no SIGEX \*\*\***

1. Tendo por base as bibliotecas de estruturas de dados apresentadas em Programação 2, implemente as funcionalidades pedidas nas duas alíneas seguintes no ficheiro **prob3.c**. Sempre que conveniente utilize as funções disponíveis nas estruturas heap e tabela de dispersão .
   1. [15 pontos] Implemente a função imprime\_capitais que lê dados de capitais mundiais a partir de um ficheiro e imprime de forma ordenada as *k* maiores capitais e o respetivo país.

void imprime\_capitais(char \*nomeFicheiro, int k)

O primeiro parâmetro da função é *string* com o nome do ficheiro e o segundo é o número de capitais a imprimir. O ficheiro contém em cada linha a seguinte informação: país capital população. O país e a capital têm apenas uma palavra.

Utilize uma *heap* para guardar de forma ordenada as capitais, de acordo com a respetiva população; utilize ainda uma tabela de dispersão para guardar as capitais e o respetivo país.

Depois de implementada a função, o programa deverá apresentar:

Toquio - Japao

Seul - Coreia\_do\_Sul

Moscovo - Russia

Teerao - Irao

Cidade\_do\_Mexico - Mexico

Jacarta - Indonesia

Lima - Peru

* 1. [10 pontos] Indique a complexidade da solução implementada na alínea anterior e uma justificação clara e sucinta (máximo 100 palavras) no comentário assinalado para o problema 3.2 no ficheiro prob3.c.

Para complementar a resposta poderá incluir comentários no problema 3.1 indicando a complexidade das respetivas instruções.

**\*\*\* Submeta o ficheiro prob3.c no SIGEX \*\*\***