Prova com consulta (Moodle). Duração: 1h25m. Parte prática [80/100 pontos]

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 árvore AVL e heap.

1.1 [20 pontos] Implemente a função procura_inicio para uma **árvore AVL** (definida pelo nó raiz) que devolve o número de *strings* que começam com um dado carater.

```
int procura_inicio(no_avl *no, char inicio)
```

O primeiro parâmetro da função é o apontador para o nó raiz da árvore e o segundo é o carater a ser usado na pesquisa. A função deverá devolver o número de *strings* da árvore que começam por inicio.

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:

```
Numero de strings comecadas por 'a': 18
Numero de strings comecadas por 'b': 19
Numero de strings comecadas por 't': 13
Numero de strings comecadas por 'z': 2
```

1.2 [20 pontos] Implemente a função seleciona_por_ordem que retorna um apontador para o k-ésimo elemento por ordem crescente de prioridade. Depois de terminar, a função deve manter todos os elementos na heap.

```
char* seleciona_por_ordem(heap *h, int k)
```

O primeiro parâmetro é o apontador para a heap e o segundo indica qual a ordem de prioridade do valor a imprimir. 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:

```
Prioridade 1: africa do sul
Prioridade 25: micronesia
Prioridade 50: moldavia
```

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

2 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 grafo e tabela de dispersão .

2.1 [25 pontos] Implemente a função distancia_cache que determina a distância de um determinado vértice de origem a outro vértice de destino. Para acelerar este cálculo deve usar uma tabela de dispersão que guarda a distância já calculada entre pares de vértices. Caso essa distância já exista na tabela de dispersão, esse valor deve ser usado, evitando a pesquisa no grafo, caso não exista, deve ser adicionado.

```
int distancia_cache(grafo *g, int origem, int destino,        tabela_dispersao *cache)
```

O primeiro parâmetro da função é o apontador para grafo, o segundo é o índice do vértice de origem, o terceiro é o índice do vértice de destino e o último parâmetro é a tabela de dispersão onde devem ser guardadas as distâncias já calculadas (cache); a distância entre os vértices é retornada pela função.

Para calcular a distância entre quaisquer dois vértices, considere a distância mais curta entre esses vértices. Poderá usar a seguinte chave para guardar a distância entre dois vértices na cache: índices dos dois vértices separados por barra ('/'), por exemplo "0/3" será a chave para a distância entre 0 e 3. Os parâmetros de entrada devem ser verificados, e a função deve retornar -1 se não for bem sucedida.

Depois de implementada a função, o programa deverá apresentar:

```
Distancia do vertice 0 a 5: 5
Distancia (na cache) do vertice 0 a 5: 5
Distancia do vertice 0 a 0: 0 OK
Distancia do vertice 0 a 3: 2 OK
Distancia do vertice 0 a 6: 6 OK
Distancia (na cache) do vertice 0 a 1: 4 OK
Distancia (na cache) do vertice 0 a 4: 1 OK
Distancia (na cache) do vertice 0 a 7: 5 OK
```

2.2 [15 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 2.2 no ficheiro prob2.c.

Para complementar a resposta poderá incluir comentários no problema 2.1 indicando a complexidade das respetivas instruções. Considere dois casos: 1) distância entre origem e destino existe na cache e 2) distância não existe na cache.

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