Miniteste 2 – Exemplo 2

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 tabela de dispersão.

1.1 Implemente a função `avl_max_subarvore` para uma **árvore AVL** que apresenta o conteúdo do nó máximo de uma subárvore. A subárvore começa no nó com chave igual ao parâmetro de entrada *inicio*:

```
void avl_max_subarvore(arvore_avl *arv, const char *inicio)
```

O primeiro parâmetro da função é o apontador para a árvore e o segundo é a chave da origem da subárvore. Os parâmetros de entrada devem ser verificados.

Depois de implementada a função, o programa deverá apresentar:

```
Arvore original: PL ID AR AF CO FR MA LS NG TH RU SE US
Maximo da sub-arvore: NG
```

1.2 Os endereços de email dos clientes loja ONLINESHOP estão organizados numa **tabela de dispersão**. O departamento tecnológico quer modificar o tamanho da tabela de dispersão devido ao crescente número de clientes. Implemente a função `tabela_copia` que copia uma tabela de dispersão para uma nova tabela com um novo tamanho mas mantendo a mesma função de *hash*:

```
tabela_dispersao* tabela_copia(tabela_dispersao *original, int
                               novotamanho)
```

Os primeiro parâmetro da função é o apontador para a tabela de dispersão original e o segundo é o novo tamanho. Os parâmetros de entrada devem ser verificados. A função deve retornar a nova tabela sem modificar a tabela original.

Depois de implementada a função, o programa deverá apresentar:

```
TABELA DE DISPERSAO (30 elementos)
  [19] : ["amorrisr@quantcast.com" :: ""]
  [25] : ["apalmerm@geocities.com" :: ""]
  ...
TABELA DE DISPERSAO (30 elementos)
  [ 5] : ["shawkinsn@europa.eu" :: ""]
  [18] : ["scooper5@google.com.hk" :: ""]
  ...
```

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 heap, grafo e lista.

Pretende-se analisar e obter informação sobre rotas de voos de uma determinada companhia aérea. Em ambas as alíneas é usada uma lista que contém os nomes de todos os aeroportos.

2.1 Implemente a função `proximas_n_chegadas`, que deve imprimir as primeiras n chegadas de aviões a um determinado aeroporto. A função deverá utilizar uma fila de prioridade baseada em ***heap*** para armazenar as chegadas.

```
int proximas_n_chegadas(lista *tempos, lista *origens,
                        lista *aeroportos, int n)
```

Os parâmetros da função incluem uma lista dos tempos de chegada e uma lista dos índices dos aeroportos de origem. Note-se que para cada posição correspondente das listas, *tempos* contém a hora de chegada e *origens* contém o índice (guardado como *string*) do respetivo aeroporto de origem. São também passados por parâmetro a lista contendo os nomes dos aeroportos e ainda o número n de chegadas que se pretende imprimir.

Os parâmetros de entrada devem ser verificados. A função deve retornar 1 se for bem sucedida e 0 em caso contrário. Considere que no máximo existem 25 chegadas.

Depois de implementada a função, o programa deverá apresentar:

```
1: Faro (PT)
2: Bruxelas Charleroi (BE)
3: Frankfurt-Hahn (DE)
4: Lisboa (PT)
5: Madrid (ES)
```

2.2 Implemente a função `pesquisa_destinos` que determina todos os destinos alcançáveis a partir de um aeroporto com voos direto. Para tal utiliza-se a estrutura **grafo**, em que os vértices correspondem a aeroportos e as arestas a voos diretos entre dois aeroportos.

Note que existem sempre voos de ida e volta, pelo que a implementação se baseia num grafo não-direcionado. O resultado deve ser apresentado numa lista ordenada dos nomes dos aeroportos alcançáveis.

```
lista* pesquisa_destinos(grafo *rotas, lista *aeroportos,
                         const char *origem)
```

Os parâmetros da função são o apontador para o grafo e para a lista de aeroportos, bem como o nome do aeroporto de origem a partir do qual se pretende efetuar a pesquisa. Os parâmetros de entrada devem ser verificados.

Depois de implementada a função, o programa deverá apresentar:

```
Destinos diretos a partir de Lille (FR) = 2
Marselha (FR)
Porto (PT)
```

3 Considere que se pretende estender a pesquisa de destinos (problema 2.2) para destinos alcançáveis com uma escala a partir de um aeroporto de origem. Indique no ficheiro **prob3.txt** como implementaria de forma eficiente este algoritmo, indicando a seguinte informação:

- estruturas de dados auxiliares utilizadas (se alguma)
- complexidade temporal do algoritmo (sem considerar a ordenação)
- justificação sucinta