Miniteste 2 – Exemplo (adaptado de miniteste 2010/11)

1 Tendo por base as bibliotecas de estruturas de dados apresentadas em Programação 2, implemente as funcionalidades pedidas nas três alíneas seguintes no ficheiro **prob1.c**. Sempre que conveniente utilize as funções disponíveis.

1.1 Acrescente uma nova função à biblioteca que consiste em transformar uma fila em pilha:

```
pilha* fila_transforma(fila *f)
```

O primeiro parâmetro da função é o apontador para a fila. Esta função deverá retornar uma nova pilha com os mesmo elementos da fila e de forma a que possam ser retirados pela mesma ordem. No caso de não conseguir criar a pilha deve retornar NULL.

Depois de implementada a função, o programa deverá apresentar:

```
Fila: F R O G
 (fila_transforma)
Pilha: F R O G
```

1.2 Acrescente uma nova função à biblioteca que consiste em remover todos os elementos da pilha com um determinado valor:

```
int pilha_remove(pilha *p, const char* string)
```

O primeiro parâmetro da função é o apontador para a fila e o segundo parâmetro é a *string* a remover. Todas as *strings* contidas na pilha que tenham o mesmo valor devem ser removidas, mantendo-se as restantes na mesma ordem. A função deve retornar 1 se for bem sucedida e 0 em caso contrário.

Depois de implementada a função, o programa deverá apresentar:

```
Antes de pilha_remove: P1 P2 P3 P1 P3 P4
(remove P1)
Depois de pilha_remove: P2 P3 P3 P4
(remove P3)
Depois de pilha_remove: P2 P4
```

1.3 Acrescente uma nova função à biblioteca que crie uma nova tabela de dispersão que contenha todos os elementos de uma tabela passada como parâmetro e use, como função de dispersão, a função passada como argumento:

```
tabela_dispersao* tabela_copia_todos(tabela_dispersao *td, hash_func *h);
```

O primeiro parâmetro é a tabela de dispersão de origem e o segundo a função de dispersão a usar na nova tabela. No caso de existir algum erro a função deve retornar NULL.

Depois de implementada a função, o programa deverá apresentar:

```
tabela_copia_todos():
      sem erros.
```

2 Considere a estrutura de dados árvore AVL. Utilizando essa estrutura, pretende-se imprimir os valores máximos e mínimo das leituras guardadas num ficheiro de texto:

```
int imprimirMaxMinLeituras (const char *nomeFicheiro)
```

O parâmetro da função é o nome do ficheiro. A função deve devolver 1 no caso de ser bem sucedida e 0 em caso contrário.

Depois de implementada a função, o programa deverá apresentar:

```
Mínimo: 11091
Máximo: 99874
```

Implemente também uma função que imprima todos os valores inferiores a 15000 guardados na árvore AVL. Use a seguinte função, implementada de forma recursiva em pré-ordem:

```
void imprimirFiltro(no_avl *no)
```

Depois de implementada a função, o programa deverá apresentar:

```
14250
11521
11091
14094
12546
14458
14310
```