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 grafo.

1.1 Implemente a função `avl_maiorstring` para uma **árvore AVL** (definida pelo nó raiz) que devolve a maior *string* guardada nos nós da árvore.

```
char* avl_maiorstring(no_avl *no)
```

O parâmetro da função é o apontador para o nó raiz da árvore e o retorno é a *string* de maior comprimento. Por comprimento entende-se o número total de caracteres.

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:

```
Maior string: republica centro-africana
```

1.2 Implemente a função `grafo_maisdistante` que determina qual o vértice de um **grafo** mais distante de um determinado vértice de origem.

```
int grafo_maisdistante(grafo *g, int origem, int *distancia)
```

O primeiro parâmetro da função é o apontador para grafo, o segundo é o índice do vértice de origem e o terceiro deverá ser usado para devolver a distância ao vértice mais distante; o índice do vértice mais distante é retornado pela função. Para calcular a distância entre quaisquer dois vértices, considera a distância mais curta entre esses vértices.

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:

```
Mais distante do vertice 1: 2 (distancia 4)
Mais distante do vertice 6: 4 (distancia 5)
```

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 e vetor.

2.1 [25 pontos] Implemente a função `heap_ordena` que cria um novo vetor ordenado seguindo uma ordem decrescente, usando uma **heap** auxiliar. A prioridade associada a cada *string* é calculada com base nos dois primeiros caracteres; considere o seguinte exemplo para determinar a prioridade de uma *string* `str`: `prioridade = (str[0] << 8) + str[1];`

```
vetor* heap_ordena(vetor *v)
```

O parâmetro da função é o vetor contendo as *strings* a ordenar. A função deve retornar um novo vetor com as *strings* ordenadas se for bem sucedida ou NULL em caso contrário, incluindo erro nos parâmetros.

Depois de implementada a função, o programa deverá apresentar:

```
yunkai
wickendon
winterfell
...
astapor
ashford
ar noy
```

2.2 [15 pontos] Indique a complexidade da solução implementada na alínea anterior e uma justificação clara e sucinta (máximo 50 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.