**Resolva cada exercício em folhas separadas**

1. Uma MultiStack é um vetor de stacks $S_0,S_1,...,S_k$ na qual a stack $S_0$ funciona como buffer e as restantes stacks $S_i$ i=1..k-1 podem acomodar no máximo 3×i elementos, estando as stacks preenchidas por ordem crescente do índice i, de tal modo que se a MultiStack possui k stacks, as k-1 primeiras stacks estão totalmente preenchidas, só a última poderá não estar. Este sistema MultiStack é usado numa empresa fabril para coleta de peças individuais e peças de montagem. Simplificadamente, para ambos os tipos de peças é registado o código da peça, para as peças individuais o tipo de embalagem necessário para acomodar a peça e para as peças de montagem o peso da peça.
   a) Defina as classes indicando apenas os atributos e a assinatura dos métodos necessários à modelação de uma aplicação que permita gerir o sistema MultiStack desta empresa fabril, com capacidade máxima k=50 stacks.
   b) Escreva uma função que calcule o peso total das peças de montagem existentes na MultiStack.

2. A inserção de peças na MultiStack com k stacks faz-se sempre na stack $S_0$, com capacidade para k peças, a qual ficando cheia todos as suas peças são transferidas para as demais stacks da MultiStack. A remoção de um código de peça na MultiStack pode envolver a remoção de várias peças na mesma stack e/ou em várias stacks, pelo que após a remoção é necessário transferir peças da(s) última(s) stack(s) preenchida(s) para as stacks envolvidas na remoção, de modo a garantir que todas as stacks estão preenchidas por ordem crescente do índice i. Escreva o método de remoção na MultiStack.

INSTITUTO POLITÉCNICO DO PORTO

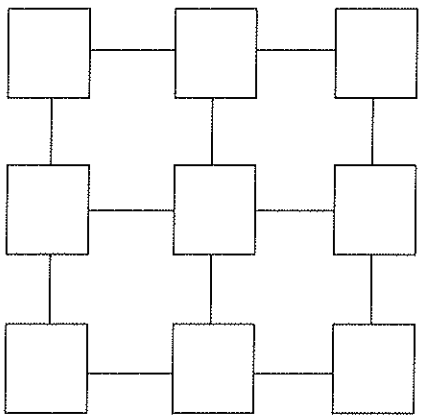
3. O método seguinte verifica se um número é capicua, ou seja, se o número é igual lido da esquerda para a direita e da direita para a esquerda, exemplo `capicua(1221,1221,0)`.

```
bool capicua (int num, int aux, int cap)
{
  if (aux == 0)
      return (num == cap) ;
  else
    {
       cap = cap * 10 ;
       cap = cap + (aux%10) ;
       aux = aux / 10 ;
       capicua (num,aux,cap) ;
    }
}
```

Faça a análise a complexidade temporal T(n) do método. Justifique.

**Resolva cada exercício em folhas separadas**

4. Cada vez mais os processadores são equipados com múltiplos núcleos (*multicore*) sendo que as interligações entre os núcleos são em malha, como apresentado na figura.



Cada núcleo tem associado uma capacidade de processamento de *stream* de dados em *bytes/seg* e é identificado pelas suas coordenadas x e y na malha. Para além disto um multicore possui ligações entre os núcleos com diferentes larguras de banda, também em *bytes/seg*.

a) Pretende-se representar de forma simplificada um caso particular destes processadores, para tal defina a classe não template MultiCore recorrendo à classe ListAdjGrafo<TV,TR>. Indique apenas os atributos e a assinatura dos métodos.

b) Implemente uma função que devolve a largura de banda da ligação entre dois núcleos diretamente ligados. Caso não estejam diretamente ligados devolve zero.

c) Implemente um método que, dado um bloco de dados em bytes e dois núcleos do *multicore*, apresenta a lista de núcleos a que corresponde o tempo mínimo de processamento entre os dois núcleos. Considere apenas o tempo de processamento nos vários núcleos, incluindo os núcleos inicial e final, despreze o tempo de transmissão entre os vários núcleos.

INSTITUTO POLITÉCNICO DO PORTO

5. Elabore um método que verifica se uma árvore binária está completamente cheia, ou seja, se todos os nós internos têm ambas as subárvores esquerda e direita.