TBL 6 - Code optimisations

Considera o seguinte programa em C, que implementa um algoritmo recursivo para calcular o número de notas positivas. O programa, dado um array de números inteiros, conta quantos são maiores do que 9.

```
1   #include <stdio.h>
2   int N=5;
3   int grades[] = {19, 6, 17, 10, 8};

4   int countGreater(int threshold, int a[], int N) {
5      int res = 0;
6      if (N>0) {
7         res = a[0] > threshold;
8         res += countGreater(threshold, a+1, N-1);
9      }
10     return res;
11  }

12  int main () {
13     int numPositives = countGreater(9, grades, N);
14     printf("Positivas: %d\n", numPositives);
15     return 0;
16  }
```

1. Interpreta e executa a função de forma a perceberes o seu funcionamento. Usa as funcionalidade disponíveis em www.onlinegdb.com.

a. Ativa um *breakpoint* na linha assinada a bold, clicando à esquerda do seu número.

b. Executa o programa em modo 'Debug' (linguagem C).

c. Na consola, usa o comando 'run' para iniciar a execução; a execução deve parar no breakpoint definido.

d. Na seção 'Display Expressions', monitoriza as expressões `a`, `a[0]`, `N` e `res`.

e. Executa o comando 'step' para avançar a execução linha a linha; atenta às secções 'Call Stack' e 'Display Expressions' à medida que a recursividade avança.

2. Analisa o código assembly da função utilizando o gcc. Usa o godbolt.org com as seguintes configurações:

- Linguagem: C
- Versão do GCC: x86-64 gcc 4.9.1
- Flags: -m32 -O0
- No menu 'Output', desativa a opção "Intel asm syntax"

3. Analisa as instruções relativas às várias invocações da função `countGreater`.

a. Com `N=5`, quantas vezes é invocada a função?

b. E se o array tiver 500 notas (`N=500`)?

4. Identifica as instruções relativas à primeira invocação da função `countGreater`, pela função `main`, e desenha o respetivo quadro de ativação de pilha (*stack frame*). Detalha os conteúdos das várias células de memória e os respectivos deslocamentos (`offsets`) em relação ao registo `ebp`.

5. Recompila o programa substituindo a flag -O0 por -Os.

a. Para além do código ter ficado mais compacto, que alteração estrutural foi introduzida pelo compilador?

b. Quantas vezes é agora invocada a função `countGreater`?

c. Por que razões terá sido feita esta reformulação?

d. Reescreve o código C de forma a aproximá-lo à versão otimizada pelo compilador.