Algoritmos e Complexidade – 6 de Novembro 2013 – Duração: 90 min

Parte A

1. Considere o programa que se segue, anotado com uma *pré-condição*, um *invariante de ciclo*, e uma *pós-condição*. Gere as respectivas condições de verificação para correcção parcial, começando por acrescentar ao código outras anotações que lhe pareçam necessárias.

```
// N >= 0
k = 0;
while ((k<N/2) && (v[k]==v[N-k-1])) {
  // (k <= N/2) /\ forall i; (0 <= i < k) ==> v[i] == v[N-i-1]
  k++;
}
if (k==N/2) r = 1;
else r = 0;
// (r = 1 /\ forall i; (0 <= i < N/2) ==> v[i] == v[N-i-1]) \/
// (r = 0 /\ exists i; (0 <= i < N/2) /\ v[i] != v[N-i-1])
```

2. Considere as seguintes definições em que a função `crescente` calcula o comprimento do *maior prefixo crescente* de um vector de inteiros, e `maxcresc` calcula o tamanho do *maior segmento crescente* de um vector de inteiros.

```
int crescente (int v[], int N) {
    int i;
    for (i=1; i<N; i++)
        if (v[i] < v[i-1])
           break;
    return i;
}
```

```
int maxcresc (int v[], int N){
    int r = 1, i = 0, m;
    while (i<N-1) {
        m = crescente (v+i, N-i);
        if (m>r) m = r;
        i=i+1;
    }
    return r;
}
```

   Para cada uma das funções, identifique o melhor e o pior caso de execução e faça a respectiva análise assimptótica do tempo de execução, com base nas comparações entre elementos do array.

3. A seguinte função recursiva calcula o número de elementos diferentes presentes num array de inteiros. Apresente duas recorrências, correspondentes ao melhor e pior casos de execução, e resolva-as, identificando em que situação ocorre cada caso.

```
int diferentes (int v[], int N) {
  int d, i;
  if (N == 0) return 0;
  d = diferentes(v+1, N-1);
  for (i=1; (i<N) && (v[i]!=v[0]); i++);
  if (i==N) return d+1 else return d;
}
```

Parte B

1. Calcule o número médio de comparações entre os elementos do array que são feitas pela função `crescente` (definida na questão 2 da Parte A). Para isso considere que os valores do array são perfeitamente aleatórios e por isso, que para qualquer índice `i`, a probabilidade de a posição `i` conter um valor menor do que a posição `i-1` é 0.5.

2. Modifique a definição da função `maxcresc` (definida na alínea 2 da Parte A) de forma a obter uma função que execute no pior caso em tempo linear no comprimento do array de entrada.