Asignatura: Fundamentos de algoritmia
Evaluación continua. Cuestionario.

Grupo A. Amarillo
Profesor: Isabel Pita.

Nombre del alumno:

1. Dadas las siguientes funciones de coste, indica su orden de complejidad más ajustado (representado por la función más sencilla) y ordena los órdenes de complejidad obtenidos utilizando $\subset$.

   a) $f_1(n) = \frac{1}{10}n + \log^2 n$
   b) $f_2(n) = \log_3 5n + \log_{10} 10n + \log_2 n^2$
   c) $f_3(n) = n \log n + n$
   d) $f_4(n) = 2^n + 3^n + 4^n + n^n$
   e) $f_5(n) = \log_3 10$
   f) $f_6(n) = n \log_2^2 n + n \log_5 n + \log_{10} n$

   Respuesta:

   a) $f_1(n) = \frac{1}{10}n + \log^2 n \in \mathcal{O}(n)$.
   b) $f_2(n) = \log_3 5n + \log_{10} 10n + \log_2 n^2 \in \mathcal{O}(\log n)$.
   c) $f_3(n) = n \log n + n \in \mathcal{O}(n \log n)$.
   d) $f_4(n) = 2^n + 3^n + 4^n + n^n \in \mathcal{O}(n^n)$.
   e) $f_5(n) = \log_3 10 \in \mathcal{O}(1)$.
   f) $f_6(n) = n \log_2^2 n + n \log_5 n + \log_{10} n \in \mathcal{O}(n \log^2 n)$.

   $\mathcal{O}(1) \subset \mathcal{O}(\log n) \subset \mathcal{O}(n) \subset \mathcal{O}(n \log n) \subset \mathcal{O}(n \log^2 n) \subset \mathcal{O}(n^n)$.

2. Dada la siguiente función:

```
int f (tMatriz const& m) {
  int suma = 0; int i = 0, j = 0;
  while (i < m.size()) {
    suma += m[i][j];
    if (j == m[0].size()-1) {++i; j = 0;}
    else ++j;
  }
}
```

   a) Indica el número de vueltas que da el bucle y el coste de cada vuelta, explicando tu respuesta.
   b) Indica su orden de complejidad.

   Respuesta.

   a) El bucle da `m.size() * m[0].size()` vueltas, porque la variable `i` solo se incrementa cuando la variable `j` llega a `m[0].size()`. El coste de cada vuelta es constante porque todas las instrucciones del bucle son de coste constante.
   b) El orden de complejidad es $\mathcal{O}(m.size() * m[0].size())$, es decir el número de filas por el número de columnas de la matriz.

3. Dado el siguiente algoritmo:

```
int f (vector<int> const & v) {
  int sum = 0;  int cont = 0;
  for (int i=0; i<v.size(); ++i)
    sum += v[i];
  for (int i =  0; i < v.size(); ++i)
    if (sum == v[i]) ++cont;
  return cont;
}
```

a) Calcula su función de coste, justificando brevemente la respuesta.

b) Indica su orden de complejidad.

Respuesta:

a) El primer bucle da `v.size()` vueltas, en cada vuelta se ejecutan una asignación, dos sumas y una comparación, en total 4 operaciones de coste constante.
El segundo bucle da `v.size()` vueltas y en cada vuelta se ejecutan dos comparaciones y dos incrementos, en total 4 operaciones de coste constante.
Hay además 4 asignaciones que se ejecutan una vez y una operación de retorno, todas ellas de coste constante.
La función de coste es: 4*`v.size()` + 4*`v.size()` + 5 = 8*`v.size()` + 5

b) El orden de complejidad es lineal respecto al tamaño del vector.

4. Escribe una expresión que cuente el número de valores positivos de un vector que se encuentran en posiciones pares.

   Respuesta:

   $positivos(v) = \#k : 0 \leq k < v.size \wedge k\,\%2 == 0 : v[k] > 0$

5. Escribe un predicado $diferentes(v)$ que exprese que todas las componentes de un vector v son diferentes.

   Respuesta:

   Existen muchas formas de expresarlo, algunas son:

   - $\forall k1, k2 : 0 \leq k1 < k2 < v.size() : v[k1] \neq v[k2]$.
   - $\forall k1, k2 : 0 \leq k1 < v.size() \wedge 0 <= k2 < v.size() \wedge k1 \neq k2 : v[k1] \neq v[k2]$.
   - $\forall k : 0 <= k < v.size() : ((\#x : 0 <= x < v.size() : v[k] == v[x]) == 1)$.

6. Dada la siguiente especificación

   P: { $v.size > 1$ }

   xxx (vector<int> v) dev int r

   Q:{ $r =$ `max` $u : 0 \leq u < v.size \wedge u\,\%2 == 1 : v[u]$ }

   Explica por qué en la precondición se exige que el vector tenga al menos dos elementos.

   Respuesta:

   El problema pide calcular el máximo de las componentes impares del vector. Para que exista un valor máximo el vector debe tener al menos una componente con índice impar.

7. Especifica un algoritmo que dados dos vectores no vacíos de números enteros, compruebe que todos los valores del primer vector son menores que los valores del segundo.

   Respuesta:

   Nos piden una especificación, por lo tanto debemos dar una precondición, la cabecera de la función donde figuren los valores de entrada y de salida y una postcondición.

   P:{ $v1.size > 0 \wedge v2.size > 0$ }

   mayores (vector<int> v1, vector<int> v2) dev bool ok

   Q { $ok \equiv \forall\, k1, k2 : 0 \leq k1 < v1.size \wedge 0 \leq k2 < v2.size : v1[k1] < v2[k2]$ }

8. Dada la siguiente especificación

   P:{ $v.size() \geq 0$ }

   xxx (vector<int> v) dev int l

   Q: { $l = \texttt{max}\ p, q : 0 \leq p \leq q \leq v.size() \wedge creciente(v, p, q) : q - p)$ }

   donde: $creciente(v, a, b) == \forall\ k1, k2 : a <= k1 < k2 < b : v[k1] < v[k2]$

   Dado el siguiente vector de entrada 1 3 5 4 2 3 5 7 9 0 2 4 5 1 7 5. Si el programa es correcto indica qué valor tomará la variable l al terminar la ejecución. Justifica tu respuesta.

   Respuesta:

   En la postcondición se especifica que l es la longitud del segmento máximo creciente del vector. En el vector que nos dan hay un segmento máximo creciente (2 3 5 7 9) que empieza en la posición 4 y tiene longitud 5.

   Por lo tanto, l = 5.