Examen de Programación 3 y III (14/07/2009)

Instituto de Computación, Facultad de Ingeniería

1. Este examen dura 4 horas y contiene **2** carillas. El total de puntos es 100.
2. En los enunciados llamamos C* a la extensión de C al que se agrega el operador de pasaje por referencia &, y las sentencias *new*, *delete* y el uso de *cout* y *cin*.
3. NO se puede utilizar ningún tipo de material de consulta. Puede usarse todo lo visto en el teórico, práctico y laboratorio sin demostrarlo, indicando claramente lo que se está usando.
4. No se contestaran dudas durante la última media hora.

Se requiere:

- Numerar todas las hojas e incluir en cada una el nombre y la cédula de identidad.
- Utilizar las hojas de un sólo lado y escribir con lápiz, iniciando cada ejercicio en hoja nueva.
- Poner en la primera hoja la cantidad de hojas entregadas, y un **índice** indicando en qué hoja se respondió cada problema.

Ejercicio 1. (35 puntos)

1. Para cada afirmación indique si es verdadera o falsa, en caso de ser verdadera demuestre sin utilizar la regla del límite, y en caso de ser falsa proporcione un contraejemplo.

1.1. $$\left.\begin{array}{l} f(n) \in O(h(n)) \\ g(n) \in O(h(n)) \end{array}\right\} \Rightarrow f(n) \in \Theta(g(n))$$

1.2. $$\left.\begin{array}{l} f(n) \in O(h(n)) \\ g(n) \in \Omega(h(n)) \\ g(n) \in O(f(n)) \end{array}\right\} \Rightarrow f(n) \in \Theta(g(n))$$

1.3. $$\left.\begin{array}{l} f(n) \in O(g(n)) \\ h(n) \in \Theta(g(n)) \end{array}\right\} \Rightarrow h(n) \in \Omega(f(n))$$

2. Dado el siguiente algoritmo:

```
bool funcion (int* a, int i, int j){                                    (1)
      int corte;                                                        (2)
      if (i == j)                                                       (3)
            return calcular(a, i, i);                                   (4)
      else {                                                            (5)
            if (i < j){                                                 (6)
                  corte = (i+j-1)/3;                                    (7)
                  if(funcion(a, i, corte))                              (8)
                        return calcular(a, i, corte);                   (9)
                  else                                                 (10)
                        if(funcion(a, corte + 1, j))                   (11)
                              return calcular(a, corte + 1, j);        (12)
                        else {                                         (13)
                              return calcular(a, i, corte);            (14)
                        }
            }
      }
}
```

La función calcular tiene el siguiente encabezado: bool *calcular(int* a, int i, int j)*
Ambas funciones reciben como parámetro de entrada un arreglo *a* y dos índices *i* y *j* que representan que en esa invocación se consideran los elementos ubicados entre los lugares *i* y *j* del arreglo *a*.
Inicialmente se invoca a *funcion* con los siguientes parámetros: *funcion(a, 1, n)*

El procedimiento *calcular* es quién realiza las operaciones básicas, y la cantidad de dichas operaciones es:

- $(3j-i+1) + j^2$ si $i \neq j$.
- 3 si i = j

2.1. Analice las distintas alternativas en un paso genérico del algoritmo e indique la cantidad de operaciones básicas en cada alternativa.
Indique en cuales de las situaciones anteriores se da el peor caso.

2.2. Plantee la recurrencia que calcula la cantidad de operaciones básicas que se realizan en el peor caso definido en el punto anterior.

Ejercicio 2. (30 puntos)

Un delivery ofrece *n* menús diferentes, donde de cada menú se conoce: su precio, su contenido en proteínas y su contenido en calorías.

Dado un cliente que quiere mantener una dieta, se quiere hallar el valor calórico más bajo posible eligiendo algunos de los *n* menús, tal que el precio total de dicha elección no supere un presupuesto M, y a su vez, el valor proteico sea como mínimo W.

Téngase en cuenta que no se pueden repetir menús.

Asuma que se tienen definidas las siguientes funciones:

- p(k) indica el precio del k-ésimo menú con $1 \leq k \leq n$.
- q(k) indica el contenido en proteínas del k-ésimo menú con $1 \leq k \leq n$.
- c(k) indica el contenido en calorías del k-ésimo menú con $1 \leq k \leq n$.

Se pide:

Dar una fórmula **recursiva** para solucionar el problema. Llame a la fórmula recursiva *f*. Explicar que representa cada paso base y cada paso recursivo de la solución, como también que representa la función *f* indicando sus índices.

Sugerencia: tenga en cuenta para definir el/los caso/s base el hecho que no hayan mas menú disponibles, discriminando según el valor proteico sea mayor ó menor igual a cero.

Ejercicio 3. (35 puntos)

Un chofer de una camioneta es contratado por una escuela para que pase a buscar a un conjunto de niños y los entregue a la misma. El recorrido que hace todos los días el chofer consta de ir desde su casa a la escuela pasando por la casa de todos los escolares que viajan en la camioneta.
Además, el recorrido debe hacerse en menos de 1 hora ya que sino los escolares llegarán tarde a la escuela.

Dado que el chofer desea entregar a los niños en hora a la escuela gastando la menor nafta posible, se desea minimizar la distancia recorrida logrando entregar a los niños en hora.

Se pide:
a) Formalizar el problema en términos de Backtracking.
Indicar: forma de la solución, restricciones explícitas e implícitas, predicados de poda y función objetivo.

NOTA: No se corregirá la parte b) si no se realizó **satisfactoriamente** la parte a)
b) Implementar una función en C* que resuelva el problema utilizando **Backtracking**.

Se dispone de las siguientes estructuras:

- tiempo[0..n+1][0..n+1], donde tiempo[i][j] indica el tiempo (en minutos) requerido para ir desde la casa del escolar i hasta la casa del escolar j.
- distancia[0..n+1][0..n+1], donde distancia[i][j] indica la distancia entre la casa del escolar i y la del escolar j.
- En ambos casos, 0 representa la casa del conductor y n+1 la escuela.