Examen de Programación 3 y III (26/02/2009)

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. (36 puntos)

a) **(16 puntos)** Considere el siguiente algoritmo:

```
bool f1(int m, double* A){

    bool ret = true;
    for(int i=1; i<m; i++){
        if f2(m,A){
            ret = ret && f1(m-1, A);
        }
    }
    return ret;
}
```

Suponga que f2(…) no modifica el 2º parámetro, no invoca a f1(…) ni contiene el operador "**&&**".

1. Indique en qué condiciones se dan el mejor y peor caso considerando el operador "**&&**" como operación básica.
2. Indique en qué condiciones se dan el mejor y peor caso considerando la función **f2**(…) como operación básica.
3. Considere **f2**(…) como operación básica. Dé una expresión de $F_W(m)$ que sea la mejor cota inferior del costo de ejecución en el peor caso cuya expresión sólo dependa de m.
4. ¿Existe un código funcionalmente equivalente cuyo tiempo de ejecución *t1* cumple $t1(m) \in \Theta(m)$? Considere que f2 tiene como tiempo de ejecución *t2* tal que $t2(m) \in O(1)$. Justifique.

b) **(20 puntos)** Demuestre las siguientes proposiciones:

b1) Sin utilizar la regla del límite

1. $f \in \Theta(g) \Leftrightarrow g \in \Theta(f)$
2. $(n+a)^b \in \Theta(n^b)$ $\quad a,b \in N^+$
3. $h_1(n) \le f(n) \le h_2(n) \forall n \ge n_0$, $h_1(n) \in \Omega(g(n))$, $h_2(n) \in O(g(n))$ $\Rightarrow f(n) \in \Theta(g(n))$
4. $\lim \frac{f(n)}{g(n)} = L \wedge 0 < L < n \Rightarrow f(n) \in \Theta(g(n))$

b2) Utilizando únicamente las definiciones de O y Ω .

1. $n \log(n) \in O(n^2)$
2. $n^5 2^n \in \Omega(n^{10})$

Ejercicio 2. (32 puntos)

Un estudiante de la carrera Ingeniería en Computación, luego de haber obtenido un número significativo de créditos, comienza a estudiar qué asignaturas electivas puede tomar.
Mirando los créditos que tiene cada una de éstas, el estudiante se propone terminar su carrera (obtener 450 créditos) tomando la menor cantidad de electivas posible. Por lo tanto, quiere encontrar el menor conjunto de electivas, tal que sumando los créditos de cada una de éstas a los créditos que él estudiante ha obtenido, llegue al menos a 450.

Según las aprobaciones que el estudiante tiene hasta el momento, puede tomar electivas sólo de dos áreas, Matemática y Programación. Dado que el estudiante está un poco cansado de la Matemática, en su búsqueda prefiere tomar la mayor cantidad de electivas del área de Programación. Por lo tanto, en caso de encontrar más de una posibilidad, elegirá aquella que agregue más electivas del área de Programación.

Se pide:

1. **(12 puntos)** 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)

2. **(20 puntos)** Implementar una función en C* que resuelva el problema utilizando **Backtracking**. La función debe retornar una tupla con los códigos de las asignaturas electivas que forman la mejor combinación para el estudiante. Indicar las porciones de código que se corresponden con las diferentes partes de la formalización.

   La función debe tener el siguiente encabezado:

   ```
   Tupla electivasMejorOpcion(int creditosEstudiante, int* codElectivas);
   ```

   - `'creditosEstudiante'` es la cantidad de créditos obtenidos por el estudiante hasta el momento y es un valor menor estricto a 450.
   - `'codElectivas'` es un arreglo de largo N y contiene los códigos de las electivas que el estudiante puede tomar (de áreas Matemática y Programación).
   - 'Tupla' es una estructura de datos definida por usted para representar una tupla.
   - Para poder identificar qué códigos son de electivas del área de Matemática y cuales son del área de Programación, suponga que los códigos de las electivas del área de Matemática son menores que 100, y los de las asignaturas del área de Programación son mayores que 100.
   - Se dispone de una función que dado un código de asignatura devuelve la cantidad de créditos de la misma, con el siguiente encabezado:

     ```
     int creditosAsignatura(int codAsignatura);
     ```

Ejercicio 3. (32 puntos)

Sean n empresas diferentes y *h* una función monótona creciente, tal que $h(x,i)$ representa el interés que se obtiene si se invierte el importe *x* en la empresa *i* con $0 \le i \le n-1$ y $x > 0$ con $x \in N$.

Dado un inversor, se quiere hallar el interés óptimo total si se tiene como cantidad de dinero a invertir *d* pesos.

Se pide:

a) **(15 puntos)** Dar una fórmula **recursiva** para solucionar el problema. Llame a la fórmula recursiva *f*.
b) **(10 puntos)** Utilizando **Programación Dinámica** implemente una función **ITERATIVA** en C* que solucione el problema de la *parte a*.
c) **(7 puntos)** Describa que cambios tendría que hacer para no hallar solamente el interés óptimo total, sino también **la cantidad de dinero que se debe invertir en cada empresa** para maximizar los intereses obtenidos si se tiene como cantidad de dinero a invertir *d* pesos. **NO es necesario implementar los cambios.**