Algoritmos y Estructuras de Datos II – 22 de Febrero de 2021
Examen Final Teórico-Práctico

Alumno: ..... Email: .....

Siempre se debe explicar la solución. Una respuesta correcta no es suficiente si no viene acompañada de una justificación lo más clara y completa posible. Los algoritmos no deben escribirse utilizando código c o de bajo nivel, sino el código de la materia y evitando la utilización innecesaria de punteros. La no observación de estas recomendaciones resta puntaje.

1. (Algoritmos voraces) Te vas n días de vacaciones al medio de la montaña, lejos de toda civilización. Llevás con vos lo imprescindible: una carpa, ropa, una linterna, un buen libro y comida preparada para m raciones diarias, con $m > n$. Cada ración i tiene una fecha de vencimiento v_i, contada en días desde el momento en que llegás a la montaña. Por ejemplo, una vianda con fecha de vencimiento 4, significa que se puede comer hasta el día número 4 de vacaciones inclusive. Luego ya está fuera de estado y no puede comerse.

   Tenés que encontrar la mejor manera de organizar las viandas diarias, de manera que la cantidad que se vencen sin ser comidas sea mínima. Deberás indicar para cada día j, $1 \leq j \leq n$, qué vianda es la que comerás, asegurando que nunca comas algo vencido.

   Se pide lo siguiente:

   (a) Indicar de manera simple y concreta, cuál es el criterio de selección voraz para construir la solución?

   (b) Indicar qué estructuras de datos utilizarás para resolver el problema.

   (c) Explicar en palabras cómo resolverá el problema el algoritmo.

   (d) Implementar el algoritmo en el lenguaje de la materia de manera precisa.

2. (Backtracking) Te vas de viaje a la montaña viajando en auto k horas hasta la base de un cerro, donde luego caminarás hasta el destino de tus vacaciones. Tu auto no es muy nuevo, y tiene un stereo que solo reproduce cds (compact-disks). Buscás en tu vasta colección que compraste en los años 90 y tenés p cds, con $p > k$, que duran exactamente una hora cada uno. Encontrás también un cuaderno donde le diste una puntuación entre 1 y 10 a cada cd de tu colección. Cuanto mayor la puntuación, más es el placer que te da escucharlo. Dado que no sos tan exigente, querés que el puntaje promedio entre dos discos consecutivos que escuches, no sea menor a 6. Así por ejemplo si en la hora 2 escuchás un cd que tiene puntaje 8, en la hora 3 podrías escuchar uno que tenga puntaje al menos 4.

   Encontrar una combinación de cds para escuchar en las k horas de viaje, cumpliendo la restricción de que en dos horas consecutivas el puntaje promedio de los dos discos sea mayor o igual a 6, maximizando el puntaje total de los k discos que escucharás.

   Se pide lo siguiente:

   (a) Especificá precisamente qué calcula la función recursiva que resolverá el problema, indicando qué argumentos toma y la utilidad de cada uno.

   (b) Da la llamada o la expresión principal que resuelve el problema.

   (c) Definí la función en notación matemática.

3. Para cada uno de los siguientes algoritmos determinar **por separado** cada uno de los siguientes incisos.

   (a) ¿Qué hace? ¿Cuáles son las precondiciones necesarias para haga eso?

   (b) ¿Cómo lo hace?

   (c) El orden del algoritmo, analizando los distintos casos posibles.

   (d) Proponer nombres más adecuados para los identificadores (de variables, funciones y procedimientos).

```
proc q(in/out a : array[1..N] of int, in x : nat)
    for j:= 1 to x do
        m:= j
        for k:= j+1 to x do
            if a[k] < a[m] then m:= k fi
        od
        swap(a,j,m)
    od
end proc
proc p(in/out a : array[1..N] of int, in i : nat)
    q(a, i-1)
    r(a, i+1)
end proc
```

```
proc r(in/out a : array[1..N] of int, in y : nat)
    for j:= y to n do
        m:= j
        while m > y ∧ a[m] < a[m-1] do
            swap(a,m,m-1)
            m:= m-1
        od
    od
end proc
```

4. Dada la especificación del tad Cola:

```
spec Queue of T where

constructors
    fun empty_queue() ret q : Queue of T
    {- crea una cola vacía. -}

    proc enqueue (in/out q : Queue of T, in e : T)
    {- agrega el elemento e al final de la cola q. -}

operations
    fun is_empty_queue(q : Queue of T) ret b : Bool
    {- Devuelve True si la cola es vacía -}

    fun first(q : Queue of T) ret e : T
    {- Devuelve el elemento que se encuentra al comienzo de q. -}
    {- PRE:  not is_empty_queue(q) -}

    proc dequeue (in/out q : Queue of T)
    {- Elimina el elemento que se encuentra al comienzo de q. -}
    {- PRE:  not is_empty_queue(q) -}
```

Implementá los constructores y operaciones del TAD utilizando la siguiente representación, donde N es una constante de tipo nat:

```
implement Queue of T where

type Queue of T = tuple
                      elems : array[0..N-1] of T
                      size : nat
                  end tuple
```

5. (Para alumnos libres) En un ABB cuyos nodos poseen valores entre 1 y 1000, interesa encontrar el número 363. ¿Cuáles de las siguientes secuencias NO puede ser una secuencia de nodos examinados según el algoritmo de búsqueda? ¿Por qué?

   (a) $2, 252, 401, 398, 330, 344, 397, 363.$

(b) $924, 220, 911, 244, 898, 258, 362, 363$.

(c) $925, 202, 911, 240, 912, 245, 363$.

(d) $2, 399, 387, 219, 266, 382, 381, 278, 363$.

(e) $935, 278, 347, 621, 299, 392, 358, 363$.