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.

Realizar los ejercicios EN HOJAS SEPARADAS.

1. (Algoritmos voraces) Se conoce el valor actual $v_1^0, \ldots, v_n^0$ de las acciones de n empresas. Afortunadamente se dispone de una "bola de cristal" que permite conocer el valor que tendrán las acciones durante los m días subsiguientes. Es decir, los valores $v_1^1, \ldots, v_1^m$ que tendrán las acciones de la empresa 1 dentro de 1 día, ..., dentro de m días respectivamente; los valores $v_2^1, \ldots, v_2^m$ que tendrán las acciones de la empresa 2 dentro de 1 día, ..., dentro de m días respectivamente, etcétera. En general, v_i^j es el valor que tendrá una acción de la empresa i dentro de j días. Estos datos vienen dados en una matriz $V[0..n, 0..m]$.

   Dar un algoritmo voraz que calcule el máximo dinero posible a obtener al cabo de los m días comprando y vendiendo acciones, a partir de una suma inicial de dinero D.

   Se asume que siempre habrá suficientes acciones para comprar y que no se cobra comisión alguna por la compra y venta. También se asume que puede comprar una fracción de acción si no le alcanza para comprar una entera, en este caso la ganancia es proporcional a la fracción que se compró. Recordar que no siempre las acciones incrementan su valor.

   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) Dados n objetos de peso $p_1, \ldots, p_n$ y m cajas de capacidad $q_1, \ldots, q_m$ se desea almacenar los objetos en las cajas de modo de utilizar el menor número de cajas posible y sin exceder la capacidad de ninguna de ellas.

   Resolvé el problema utilizando la técnica de backtracking dando una función recursiva. Para ello:

   - Especificá precisamente qué calcula la función recursiva que resolverá el problema, indicando qué argumentos toma y la utilidad de cada uno.
   - Da la llamada o la expresión principal que resuelve el problema.
   - 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).

```
fun s(v: nat, p: array[1..n] of nat) ret y: nat
    y:= v
    while y < n ∧ p[y] ≤ p[y+1] do
        y:= y+1
    od
end fun
```

```
fun t(p: array[1..n] of nat) ret y: nat
    var z: nat
    y, z:= 0, 1
    while z ≤ n do
        y, z:= y+1, s(z,p)+1
    od
end fun
```

```
fun u(p: array[1..n] of nat) ret v: bool
    v:= (t(p) ≤ 1)
end fun
```

4. (a) Especificá el tipo Lista de elementos de algún tipo T, indicando constructores (para crear una lista vacía o agregar un elemento al comienzo de una lista ya existente) y operaciones para:

   - indicar si una lista es vacía
   - devolver el primer elemento de la lista
   - devolver el último elemento de la lista
   - devolver la lista resultante de eliminar el primer elemento
   - devolver la lista resultante de eliminar el último elemento
   - devolver la longitud de la lista
   - agregar un elemento al final de la lista

   Además de las operaciones comunes a todos los TADs para copiar y destruir.

   (b) Implementá el tipo de datos utilizando punteros de manera que todas las operaciones menos la de obtener la longitud, como tampoco las operaciones de copia y destrución, sean de orden constante.

   (c) Utilizando el tipo **abstracto** Lista de elementos de tipo Nat implementá un procedimiento para invertir el orden de los elementos de una lista. Por ejemplo si la lista de entrada es [1,2,3,4,5], luego de llamar al procedimiento debería ser [5,4,3,2,1].

5. (Para alumnos libres) Tiene una inmensa base de registros con datos de los cientos de miles de afiliados a una obra social ordenada alfabéticamente según sus nombres. Todos los días se agregan unas decenas de nuevos afiliados. Sus registros se agregan al final, y entonces la base no queda perfectamente ordenada. Se decide correr, cada noche, un algoritmo de ordenación para restablecer el orden para el día siguiente.

   ¿Cuál/cuáles de los algoritmos de ordenación sería/n más apropiados para utilizar en este caso? ¿Por qué?