Trabajo Práctico Adicional Nº 3
Repaso para 2do Parcial

Los siguientes ejercicios fueron extraídos de parciales y recuperatorios tomados durante el período 2008-2010.

Ejercicio 1: Recursividad

Los detectives de una agencia se envían todos los mensajes cifrados por motivos de seguridad. El algoritmo que están utilizando en la actualidad consiste en intercambiar cada vocal por la letra que la precede (si existe), por ejemplo si consideramos la palabra `Uruguay` la primer letra (`U`) no se intercambia porque no existe ninguna que la preceda, la `r` no es una vocal entonces no se intercambia, la tercera letra (segunda `u`) se intercambia con la `r` (nos quedaría `Uurguay`), la `g` no se intercambia, la `u` (tercera `u`) se intercambia con la `g` (queda `Uurugay`), la a se intercambia con la que la precede que ahora es la `g` (luego del intercambio previo), por último la `y` no se intercambia. Así si la frase a codificar fuera: `Uruguay fue el mejor equipo de America.`
La frase codificada sería: `Uuruagy uefe l emojre uiqop edA emirac.`

Escribir un planteo y un **procedimiento** recursivo en Pascal que reciba un archivo de caracteres `parrafo.dat` y retorne otro archivo de caracteres `codificado.dat` con el mensaje codificado.

Ejercicio 2: Recursividad

Escriba un planteo recursivo y luego una **función** en Pascal para resolver el siguiente problema.: se desea calcular la suma de los dígitos pares de un número.
Ejemplos suma(1248) = 14; suma(137) = 0; suma(2) = 2 y suma(0) = 0.

Ejercicio 3: Recursividad Dado un archivo de caracteres, se desea mostrar por pantalla todos los caracteres del archivo pero de a N por línea y además la cantidad de líneas generadas. Por ejemplo si se desear imprimir de a 5 caracteres por línea el siguiente archivo

```
E R ( 7 & 3 y e h 0 @ 8
```

El resultado en pantalla sería

```
E R ( 7 &
3 y e h 0
@ 8

Cant. Líneas: 3
```

Escriba un **planteo recursivo** para resolver el problema. Implemente el planteo con un **procedimiento recursivo** en Pascal que reciba como uno de sus parámetros un valor natural N que se utilizará para generar cada línea.

Ejercicio 4: Manejo de Archivos y Descomposición en Subproblemas

Una empresa dispone almacenado en sus archivos información sobre posibles candidatos para ser incorporados a su staff. De cada candidato se identifica por número de registro, su apellido y un nombre (todos obligatorios), dicha información se almacena en un archivo de caracteres llamado `candidatos.dat`.

Cada candidato presenta como parte de su curriculum vitae un listado de cursos realizados, los cuales se almacenan en el archivo de enteros llamado `cursos.dat`. Dicho archivo tendrá información sólo de aquellos candidatos que presenten certificados que avalen su asistencia a los cursos, por lo tanto, puede ser que existan candidatos que no se encuentren en este archivo (`cursos.dat`) porque no presentan ningún certificado. Por cada candidato que registre cursos, se agregará la siguiente información:

- En primer lugar el numero de registro del candidato (*R*),
- luego la cantidad de cursos que registra (*N*),

- y a continuación los *N* códigos de todos los cursos que registra

Así por ejemplo, `9843 5 101 104 302 210 502` se refiere a que el candidato registro `9843` registra `5` cursos con los códigos `101 104 302 210 502`.

Se dispone luego de otro archivo de enteros llamado `puntajes.dat`, con los puntajes que la empresa asigna a cada curso. Si un curso no es de interés para la empresa, el mismo no aparecerá en el archivo. Así, por ejemplo, el archivo `puntaje.dat` podría consistir de la siguiente información `302 10 104 10 103 15 502 20`, donde establece que el curso `302` vale `10` puntos, el `104` vale `10` puntos, el `103` vale `15` puntos y el `502` vale `20` puntos. Si un curso no está en el archivo (como `101` o `210`) su puntaje es `0`.

Realice un programa que genere un archivo de caracteres llamado `resultado.dat` en el cual por cada candidato, se guarde el número de registro, su apellido y nombre, y el puntaje total que obtiene si se suman todos los puntajes de los cursos que ha asistido.

Obs: En este ejercicio se pondrá especial atención en que se divida el problema en subproblemas correctamente. Recuerde hacer los ejercicios en hojas separadas, indente el código y escriba en forma clara.

Ejercicio 5: Manejo de Archivos y Descomposición en Subproblemas

Considere que se dispone de un archivo `CURSADAS.DAT` de números enteros donde se almacenan los códigos de las materias que cursó un alumno particular. Por ej. `CURSADAS.DAT = 99 123 121 101 201 203 <eof>`

Se asume que el archivo `CURSADAS.DAT` debe contener al menos una materia, ésta será el código `99` que corresponde al curso de nivelación.

Considere un archivo de enteros `QUIEREANOTARSE.DAT` donde se almacenan los códigos de 3 materias que ese mismo alumno quiere anotarse para cursar. Por ej. `QUIEREANOTARSE.DAT = 301 302 303 <eof>`

Considere un archivo `CORRELATIVAS.DAT` donde se almacenan para cada materia su única correlativa (se asume que toda materia de la carrera tiene una única correlativa), esto es, el archivo tendrá un número par de elementos donde se alternan un código de una materia y el correspondiente código de su correlativa a continuación. Por ej.

```
CORRELATIVAS.DAT = 101 99 102 99 121 101 123 101 201 101 203 121 301 123 302
201 303 102 401 201 405 123 <eof>
```

Implemente en Pascal una solución que permita generar un cuarto archivo `ACEPTADAS.DAT` con las materias que el alumno está en condiciones de anotarse para cursar ya que tiene las correlativas. Siguiendo el ejemplo, puede cursar `301` porque tiene cursada `123`, también `302` porque cursó `201`, pero no puede cursar `303` porque su correlativa (`102`) no fué cursada por el alumno.

```
ACEPTADAS = 301 302 <eof>
```

Obs: En este ejercicio se pondrá especial atención en que se divida el problema en subproblemas correctamente. Recuerde hacer los ejercicios en hojas separadas, indente el código y escriba en forma clara.

Ejercicio 6: Traza y Entornos

```
program THREE;
const ewe = 1;
type tdog = integer;
procedure DOG(bee:real;var emu:char);
    function FOX(var dog:real):real;
    begin ... end; { FOX }
    procedure CAT(rat:tdog);
        function PIG(fox:char):real;
        begin ... end; {PIG}
        function ANT(ewe:real):boolean;
        var dog: boolean;
        begin ... end; {ANT}
    begin ... end; {CAT}
    procedure OWL(dog:boolean);
        procedure FLY(ram:integer);
            function BEE(ant:char):char;
            const dog = 3;
            begin ... end; {BEE}
        var FOX:integer;
        begin ... end; {FLY}
    var pig: integer;
    begin ... end; {OWL}
begin ... end; {DOG}
function RAT(fly:real):tdog;
begin ... end; {RAT}

function FISH(m,n : integer):integer;
var ans: integer;
begin
    if (m < 10) then
        if (n < 10) then ans := m + n
        else ans := FISH(m, n-2) + n
    else ans := FISH(m-1, n) + m;
    FISH := ans;
end;
begin
    m := 11; n := 11;
    writeln('-----',FISH(m,n)); readln
end.
```

a) ¿Qué se muestra por pantalla como resultado de la ejecución del programa *three*?

b) ¿Cuál es el alcance del identificador *dog* definido en el ambiente *three*?

c) ¿En qué ambientes de referenciamiento es visible el identificador *fox* definido en el ambiente *dog* ? Justifique su elección.

(1) `DOG-FOX-CAT-PIG-ANT-OWL-FLY-BEE`
(2) `DOG-FOX-CAT-PIG-ANT-OWL-FLY-BEE-RAT`
(3) `DOG-FOX-CAT-ANT-OWL`
(4) `DOG-RAT`
(5) `THREE-DOG-FOX-CAT-OWL`
(6) `DOG-FOX-CAT-ANT-OWL-BEE`
(7) Todo el programa.

d) ¿Qué identificadores forman parte del entorno no local de *bee*? Para cada identificador que mencione, indique en que entorno está definido y el tipo (variable, constante, función, procedimiento, etc).

Obs: Por motivos de espacio se obviaron las sentencias forward, ***asuma que todas las primitivas tienen una sentencia forward****. Además se obvió el código de algunas primitivas también por motivos de espacio. Recuerde hacer los ejercicios en hojas separadas y escriba en forma clara.*