Examen conv. 3 curso 97-98 5 Septiembre 1998 1 hora 30 minutos

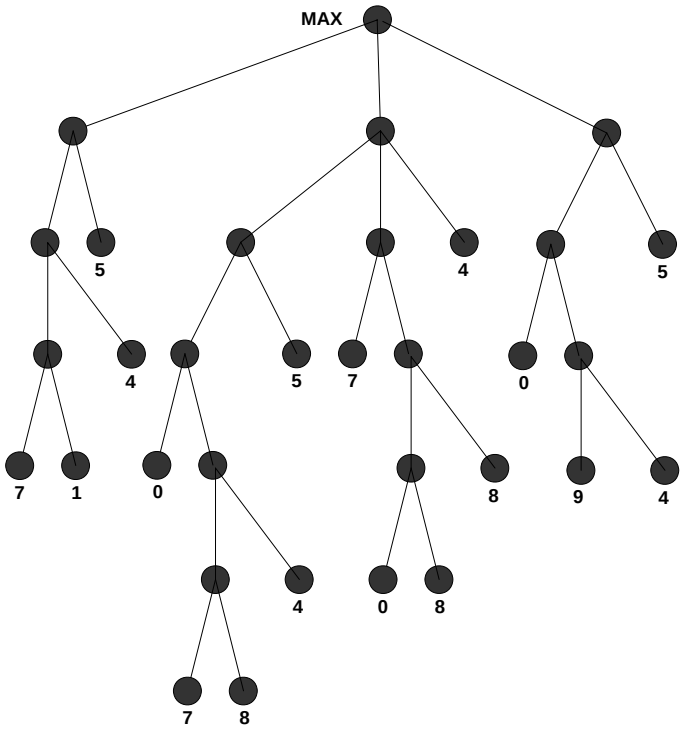
Apellidos y Nombre: .....

Problema 1: Búsquedas no informadas *[10 puntos]*

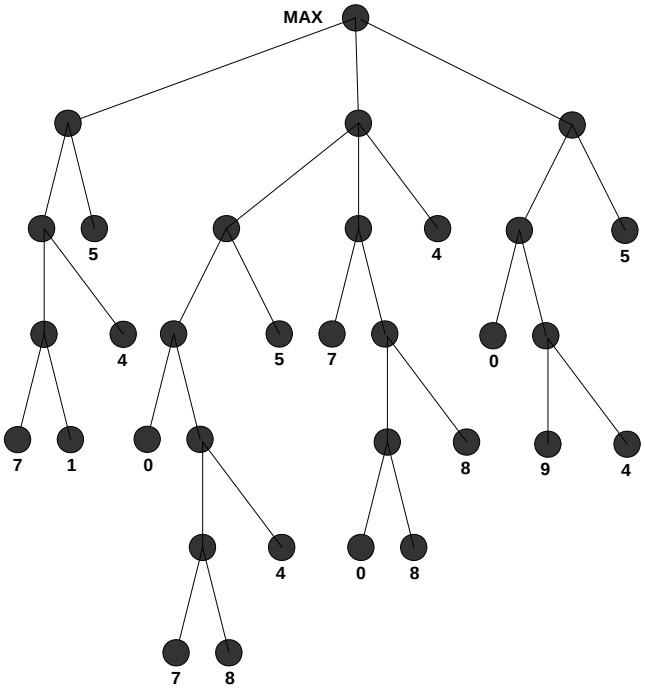
a.- Describe un espacio de búsqueda en el que la búsqueda con profundización iterativa obtiene muy peores resultados en términos del número de nodos expandidos que una búsqueda primero en profundidad (ilustra tu explicación con un dibujo del árbol de búsqueda).

b.- Describe un espacio de búsqueda en el que la búsqueda en anchura obtiene mucho mejores resultados en términos del número de nodos expandidos que una búsqueda primero en profundidad (ilustra tu explicación con un dibujo del árbol de búsqueda).

Problema 2: Búsqueda en juegos *[10 puntos]*



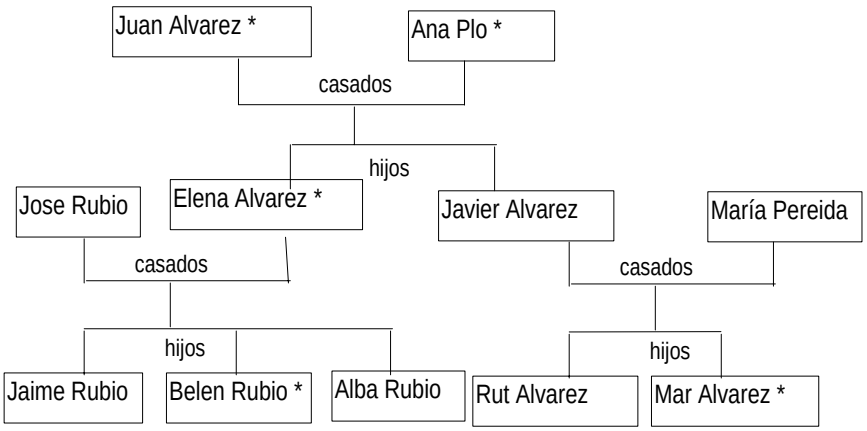
Realiza una poda alfa-beta de derecha a izquierda



Realiza una poda alfa-beta de izquierda a derecha en el ábol de la figura B.

Discute porque surgen diferentes podas.

Problema 3: Frames/Common Lisp/Prácticas *[25 puntos]*



Se desea utilizar el esquema de representación IERL utilizado en la práctica 4 (en un anexo del examen encontrarás parte del código de la práctica que te facilitará recordarlo) para representar la información mostrada en la figura anterior (los * indican que la persona ha fallecido).

Se pide que escribas el código necesario (tanto sus propios forms como los de sus antecesores en la jerarquía de herencia) para crear los forms de “Javier Álvarez” y de “Elena Álvarez”.

Crea también el código necesario (a través de las ataduras procedurales de acceso adecuadas) para que todas las personas dispongan de:

- un slot denominado “huérfanop” que informe si la persona tiene vivo alguno de los padres,
- un slot denominado “primos_vivos” que devuelva el número de primos vivos que tiene.

Al tratar de escribir en los mencionados slots debe salir un mensaje por pantalla indicando que la acción no es posible.

Problema 4: Búsquedas no informadas/Common Lisp/Prácticas *[20 puntos]*

El siguiente código corresponde al programa de búsqueda que has utilizado en las prácticas.

```
 (defun mc ()
  (setq nodos-expandidos nil                          ;(1)
        el-nodo-inicial (crea-nodo estado-inicial nil 'nodo-inicial)
        nodos-a-expandir (list el-nodo-inicial)       ;(2)
        )
  (do* ((el-nodo (pop nodos-a-expandir)
                 (if nodos-a-expandir
                   (pop nodos-a-expandir)             ;(4)
                   (return (mendaje-de-fin))))      ;(3)
        )
       ((objetivop el-nodo)                           ;(5)
        (escribe-solucion el-nodo)
        )
    (push el-nodo nodos-expandidos)
    (reorganizar-nodos-a-expandir                     ;(7)
      (expandir-nodo el-nodo))))                      ;(6)

(defun objetivop (nodo) ...)

(defun expandir-nodo (nodo) {let* ... }}

(defun reorganizar-nodos-a-expandir (nodos)
  (and nodos
       (setq nodos-a-expandir (append nodos-a-expandir nodos))
      ))

..
```

Indica que modificaciones (mínimas) se deben hacer para que se realice una búsqueda con profundización iterativa (puedes indicar cambios en el propio código).

ANEXO 1

Ejemplo de utilización del esquema de representación IERL (práctica 4)

```
;;; CALCULA-RUIDO es una funcion IF-NEEDED que calcula el nivel de ruido de un apartamento.
;;; Si existe un valor = lo toma, si no lo calcula dependiendo de la altura del piso.
;;;
 (defun calcula-ruido (form slot)
  (let* ((valor (find-aspect-from-supers form 'ruido-calle '=))
         piso)
    (cond (valor valor)
          (t (setq piso (get-value form 'piso))
             (cond ((> piso 15) 'muy-bajo) ((> piso 8) 'bajo) ((> piso 4) 'medio)
                   ((> piso 1) 'alto) (t 'muy-alto))))))

(defun gbd ()
  (setq  *tabla-de-forms* nil)
  (let ((la-form (create-form :is-a nil :name 'object))
        )
    (form :name 'cosa
          :is-a (find-form 'object))

    (form :name 'vivienda
          :is-a (find-form 'cosa))

    (form :name 'apartamento
          :is-a (find-form 'vivienda)
          :slots (list (list 'ruido-calle 'if-needed #'calcula-ruido)))

    (form :name 'chalet
          :is-a (find-form 'vivienda)
          :slots (list (list 'ruido-calle 'if-needed #'calcula-ruido-en-casa) ) )

    (form :name 'chalet-montesol-5
          :is-a (find-form 'chalet)
          :slots '((calle-nombre = urbanizacion-montesol)
                   (calle-numero = 5)
                   (numero-habitaciones = 5)
                   (pisos = 2)
                   (distancia-carretera = 200)))))
```

ANEXO 2

Alfa: (para un nodo MAX) es el valor más **alto** visto hasta el momento de los valores finales, calculados hacia atrás, de sus sucesores.
Beta: (para un nodo MIN) es el valor más **bajo** visto hasta el momento de los valores finales, calculados hacia atrás, de sus sucesores.

Poda (fin de la llamada recursiva)
Puede suspenderse la exploración por debajo de cualquier nodo MIN que tenga valores de Beta menores o iguales que el valor Alfa de cualquiera de sus nodos MAX ascendientes suyos.
Puede suspenderse la exploración por debajo de cualquier nodo MAX que tenga valores de Alfa mayores o iguales que el valor Beta de cualquiera de sus nodos MIN ascendientes suyos.