**Practica Calificada 1 – Sección 2**

Profesor Heider Sanchez

**Alumno:**

**Instrucciones:**

* La prueba dura 110 minutos.
* Resuelva los ejercicios en un documento diferente.
* No copie los enunciados, solo lo necesario para resolver el ejercicio.
* Mantenga el orden de los enunciados en el documento de resolución.
* Queda prohibido cualquier comunicación entre alumnos.
* **Está prohibido revisar cualquier página web que no sea Canvas.**
* **Sino sabe la respuesta, deje la pregunta en blanco.**

1. **(8 pts) Responder las preguntas en el Kahoot**
2. **(2 pts) Registros de Longitud Fija: Eliminación**

Dada la siguiente tabla de datos, se le aplicar la eliminación de registros usando la técnica de: mover el ultimo registro de la tabla a la posición del registro a eliminar.

“data.dat”

|  |  |  |  |
| --- | --- | --- | --- |
| 7 registros | |  |  |
|  | Id | Nombre | Ciclo |
| 1 | P-102 | Andrea | 5 |
| 2 | P-150 | Carlos | 7 |
| 3 | P-362 | Cinthya | 3 |
| 4 | P-931 | Josimar | 5 |
| 5 | P-187 | Jorge | 1 |
| 6 | P-312 | Mabel | 3 |
| 7 | P-082 | Keiko | 9 |

1. (1 pts) Muestre el estado de la tabla después de eliminar:

P-102, P-362, P-150, P-082 (en ese orden)

1. (1 pts) Muestre el estado de la tabla después de insertar:

|  |  |  |
| --- | --- | --- |
| P-390 | Peter Castle | 10 |

|  |  |  |
| --- | --- | --- |
| P-011 | Lee Yoo-mi | 1 |

1. **(4 pts) AVL File**

El AVL File es una alternativa la Sequential File, con la ventaja de gestionar el archivo de datos dinámicamente ante inserciones y eliminaciones. Al representar un árbol balanceado la complejidad de acceso siempre es O(log n).

struct Record

{

    T key;

...

    long left;

    long right;

    int height;

};

1. (3 pts) Diseñe el algoritmo de búsqueda por rango en código C++ lo más optimo posible. Asuma que tenemos un archivo de datos gigantesco y el rango de búsqueda es muy pequeño.

vector<Record> range\_search(TK begin, TK end)

1. (1 pt) ¿Es mejor o peor que aplicar búsqueda por rango usando B+ Tree?

1. **(3 pts) Dynamic Hashing**

Construya el índice hash dinámico (árbol digital) a partir de los siguientes datos:

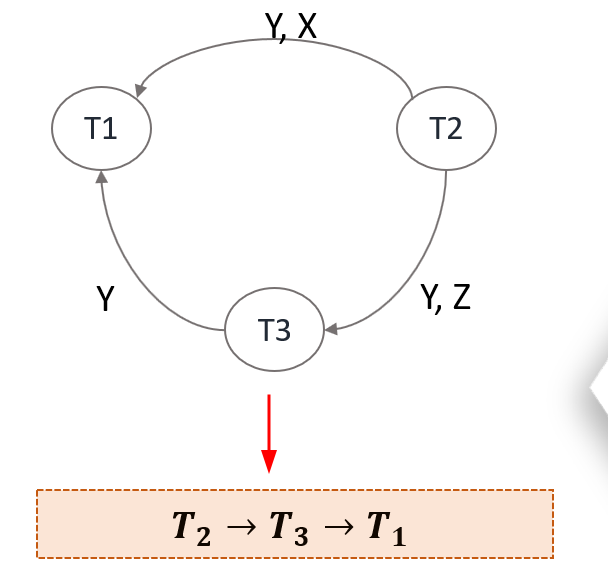
**Keys: 2, 3, 7, 8, 9, 12, 15, 18, 20, 21, 25, 33, 38, 47, 54, 61, 65**

Tome como profundidad global 𝐷=3 y un factor de bloque 𝑓𝑏=3. **Los bits se van tomando de izquierda a derecha.**

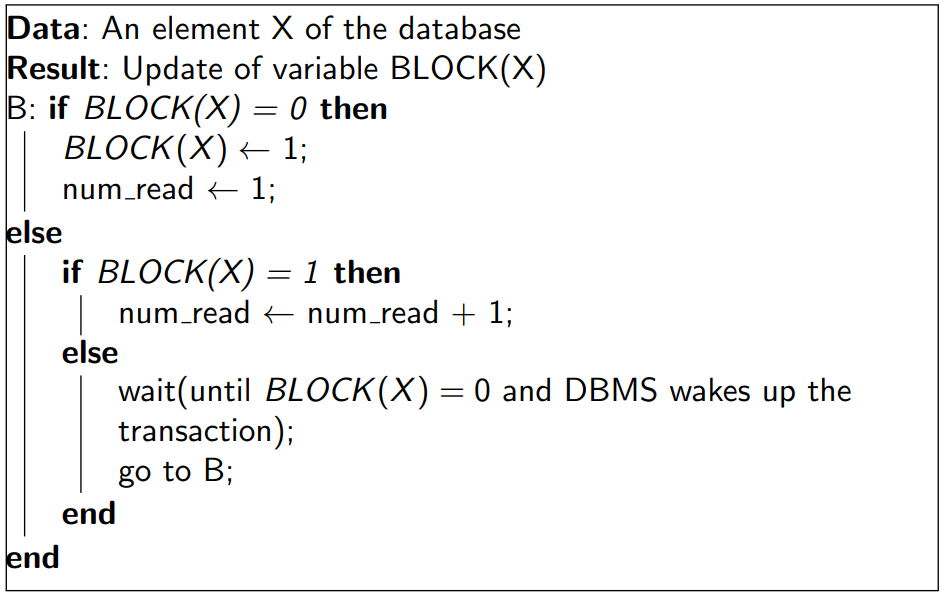
1

0

1. (1 pts) Muestre la tabla de keys y códigos binarios
2. (2 pts) ¿Cómo queda el índice? Muestre el árbol final y los buckets.
3. **(3 pts) Control de Concurrencia**
4. (1.5 pts) Diseñe un algoritmo que retorne la planificación secuencial equivalente a partir del grafo de precedencia.



1. (1.5 pts) ¿Cómo modificaría el algoritmo de bloqueo compartido para considerar también el caso cuando el recurso X está bloqueado con protocolo PU?.

****