Correccion

“Funcionalidad:

- El programa devuelve los elementos en un orden incorrecto para todos los ejemplos

Precondiciones y verificaciones: - Faltan verificaciones de la invariante y de precondiciones de las funciones.

Estructura struct s_pqueue: + La estructura principal es correcta

Estructura struct s_node: + La estructura de los nodos es correcta

Función create_node(): + Los nodos se crean de manera correcta

Función destroy_node(): + Los nodos se destruyen de manera correcta

Función pqueue_empty(): + Se crea correctamente una nueva instancia vacía de la cola

Función pqueue_is_empty(): + Se indica correctamente si la cola es vacía o no

Función pqueue_size(): + Se devuelve el tamaño de la cola correctamente en orden constante

Función pqueue_peek(): + Se devuelve el elemento más prioritario según la representación especificada

Función pqueue_peek_priority(): + Se devuelve la prioridad más alta según la representación especificada

Función pqueue_dequeue(): + Se desencola correctamente el elemento más prioritario

Función pqueue_destroy(): + La función destruye correctamente la instancia, liberando toda la memoria

Función pqueue_enqueue():

- En el caso que la cola es no vacía, la condición del `while` principal `post->priority < priority` debería ser `post->priority <= priority` ya que cuando el nuevo nodo tiene la misma prioridad que un nodo de la cola, debe agregarse después (tal como se lo hace en una cola normal).
- En el ‘if’ posterior al ciclo principal del caso no vacío, el caso `prev->priority < new_node->priority` no funciona bien si el ciclo terminó debido a que `post->next == NULL`. Solo funciona bien cuando se cumple `post->priority >= priority`.
- Si la cola es vacía el elemento se agrega correctamente
- Si la cola tiene un elemento, se encola correctamente un nuevo elemento.

- Si el nuevo nodo es más prioritario que el elemento menos prioritario de la cola, entonces se agrega correctamente.

Función invrep():

- Se verifica la propiedad fundamental de representación de la cola de prioridades, aunque queda el último nodo sin verificarse culpa de la condición `current->next != NULL`"

El mayor problema está en `pqueue_enqueue()` donde el algoritmo quedó con un par de bugs.

Cosas a tener en cuenta a futuro

- Enfocarse mas en la verificacion del invariante y de las precondiciones de las funciones
- Cuando hay problemas de orden chequear si se están usando correctamente los operadores de orden.
- Evitar usar dos punteros al iterar siempre que se pueda.
- Evitar usar **p->next != null** como condicion
- Pensar el problema y los casos antes de implementarlo

tema C

- Si hay un array de queues hay que el tipo del array es debe tener **
- Si no hay ninguna propiedad que comprobar, verificar el size
- Para iterar sobre un array de TADs conviene usar indices