Project: Persistent Todo Queue (CLI App)

Problem

You are building a simple CLI-based Todo application. The app should:

- Allow users to add tasks
- Store tasks in a FIFO queue
- Persist tasks to disk using **Borsh serialization**
- Restore tasks when the program restarts
- Allow processing (completing) tasks in order

The goal is to design a clean, generic queue system that stores serializable data and persists it safely. This simulates a very small task management backend.

Requirements

Core Functionality

Your app must support the following CLI commands:

• **Add a task**

todo add "Buy groceries"

Adds a task to the queue.

• **List all tasks**

todo list

Print all pending tasks in FIFO order.

• **Complete next task**

todo done

Removes the oldest task from the queue and marks it as completed.

• **Exit and restart**

When the program restarts, previously added tasks must still exist.

Data Modeling

You must define:

A Todo struct

```
struct Todo {
    id: u64,
    description: String,
    created_at: u64,
```

}
It must derive:

- BorshSerialize
- BorshDeserialize
- Debug
-

Generic Queue

You must implement your own generic queue:

```
pub struct Queue<T> {
    items: VecDeque<T>,
}
```

It must support:

- enqueue
- dequeue
- peek
- len
- is_empty

It must be fully generic over T.

Persistence (Serialization)

The queue must:

- Serialize to a file (e.g., todos.bin)
- Deserialize from the file on startup
- Overwrite the file whenever the queue changes

You must use borsh

No JSON, no Serde — use Borsh.