Bugs:

1. Joining a detached thread.
2. Incorrect referencing of a variable.
3. Printing count incorrectly.
4. Not locking mutex for number of producer threads variable.
5. Not locking mutex for queue when testing while condition. (resulting in incorrect printing order.)

→ Fixes in code. Details in comments.

Q1

1. Exit returns no value. But pthread_exit returns a value to a joinable thread if any.
2. Exit deallocates thread shared resources like semaphores, mutex etc. But pthread_exit does not deallocate these resources.
3. For the given code, calling exit from the main function/thread results in a call to exit() and hence the child threads are all cleaned up. When pthread_exit is called, the child threads (producer consumer) continue to run.
4. For given code, exit is used when thread creation fails which makes sense because there are no threads that can be allowed to execute.
5. For given code, pthread_exit is used when thread_join fails which makes sense because we want to allow the threads to continue execution but since they cannot join we make the main thread exit.

Q2.

The value in thread_return in main comes from the *count variable created in consumer thread. pthread_join definition says that the last argument passed to it is where the return value of the joining thread is stored. So:

- Calling/Joinable thread - main
- Argument passed to pthread_join to store return value - thread_return
- Joining thread - consumer_thread
- Return value - count

Q3.

The value in thread_return can still come from count if consumer exits using pthread_exit(&thread_return). //executed and verified this.

Q4.

If the joining thread terminates before thread_join is called then thread_join immediately returns. Not sure what happens to the return value though.

Q5.

A main thread can join another thread as far as:

- Yes, any thread can call pthread_join() as far as the thread id of the consumer/producer thread is known to the calling thread.
- the thread is joinable.
- the thread is detached from main (since chances are main created the thread in the first place :P)

Q6.

Because effective utilization of resources. Thread has no idea when the next item will be available. The wait could be indefinite. The optimal things to do is relinquish CPU to some other thread that is ready.