Question 1
Yes, the keyboard control in Lab 3 is noticeably different from the old model in Lab 1 because Lab 1 did not use a **queue** system to store multiple inputs. This means that in Lab 1, only the most recent input was registered, leading to a less responsive and potentially frustrating experience.

In a multiplayer setting, keeping the Lab 1 control mechanism without a queue would create an “unfair” situation because:

- Faster typists or players with better timing would dominate, as only the last input is registered.
- Players could not buffer multiple keypresses (reference pacman), leading to inconsistent movement.
- Input delays could make it feel like commands are ignored, leading to an unbalanced experience.

With a queue system, all keypresses are stored and processed in order, so gameplay is more fair.

Question 2
Generate food on the fly with a pseudo-random but ascending number trend without using a stack:

```
int generateItem() {
    int increment = std::rand() % 5 + 1; // Random step between 1 and 5
    lastGenerated += increment;
    return lastGenerated;
}
```

Pros of On-the-Fly Generation:

- No need to pre-generate all scores, reducing memory usage during initialization
- Reduces upfront computation time.

Cons of On-the-Fly Generation:

- More computationally expensive in real-time since the scores must be calculated during gameplay.

Question 3
Yes, Stack and Queue ADT were simpler to implement than List ADT and also required less lines of code.

I used **composition** to make the implementation easier, using SHLinkedList, which was already coded, in my Stack (refer to code).