## Potential Inventory Data Structures

The following data structures were considered for use in the Zorkish application.

### 1: array of Item

| Pros | Cons |
|---|---|
| Element access time is O(1) | Size of array must be stored and updated manually |
| Small memory footprint | Cannot be resized without creating a new array |
| Has a defined ordering | Requires manual memory management |

### 2: vector<Item>

| **Pros** | **Cons** |
|---|---|
| Fast to check if inventory contains an Item | Slow to count the number of items if inventory is allowed to contain more than one of the same item |
| Items can easily be added to the back | Items cannot be easily added elsewhere |
| Has a defined ordering | Hard to remove elements |

### 3: list<Item>

| **Pros** | **Cons** |
|---|---|
| Items can easily be added anywhere in the list | Slow to check if inventory contains an item, especially if it is near the end of the list (or the beginning if using a reverse iterator) |
| Easy to remove elements | |
| Has a defined ordering | |

### 4: map<Item,unsigned int>

| **Pros** | **Cons** |
|---|---|
| Easy to check if inventory contains an Item | Element ordering is based on the value of Item |
| Easy to check the number of a certain Item the inventory contains | Either operator < or a comparison function must be defined for Item |
| Easy to add elements | Item must define the equality operator == |
| Easy to add more than 1 of an element | Item must be immutable (otherwise there needs to be n instances of Item, rather than 1 instance and a number) |
| Easy to remove elements | |
| Easy to remove more than 1 of an element | |
| Easy to remove all elements | |

### 5: unordered_map<Item,unsigned int>

| | |
|---|---|
| Easy to check if inventory contains an Item | Elements are unordered |
| Easy to check the number of a certain Item the | A hash for Item must be provided |

| | |
|---|---|
| inventory contains | |
| Easy to add elements | Item must define the equality operator == |
| Easy to add more than 1 of an element | Item must be immutable (otherwise there needs to be n instances of Item, rather than 1 instance and a number) |
| Easy to remove elements | |
| Easy to remove more than 1 of an element | |
| Easy to remove all elements | |

The following things are true of the Zorkish inventory:

1. The player may pick up more than 1 of any given item
2. Items are immutable
3. Item ordering is irrelevant

The inventory must support the following:

1. Adding and removing items
2. Picking up multiple of the same item
3. Listing total number of items
4. Listing the number of a specific item

Given these constraints, it was chosen to use **unordered_map<Item,unsigned int>** as the basis for the inventory implementation, because 2 out of 4 cons are irrelevant based on constraints, and Item already defines the equality operator. In addition, the list of pros lines up with the desired functionality of the inventory.