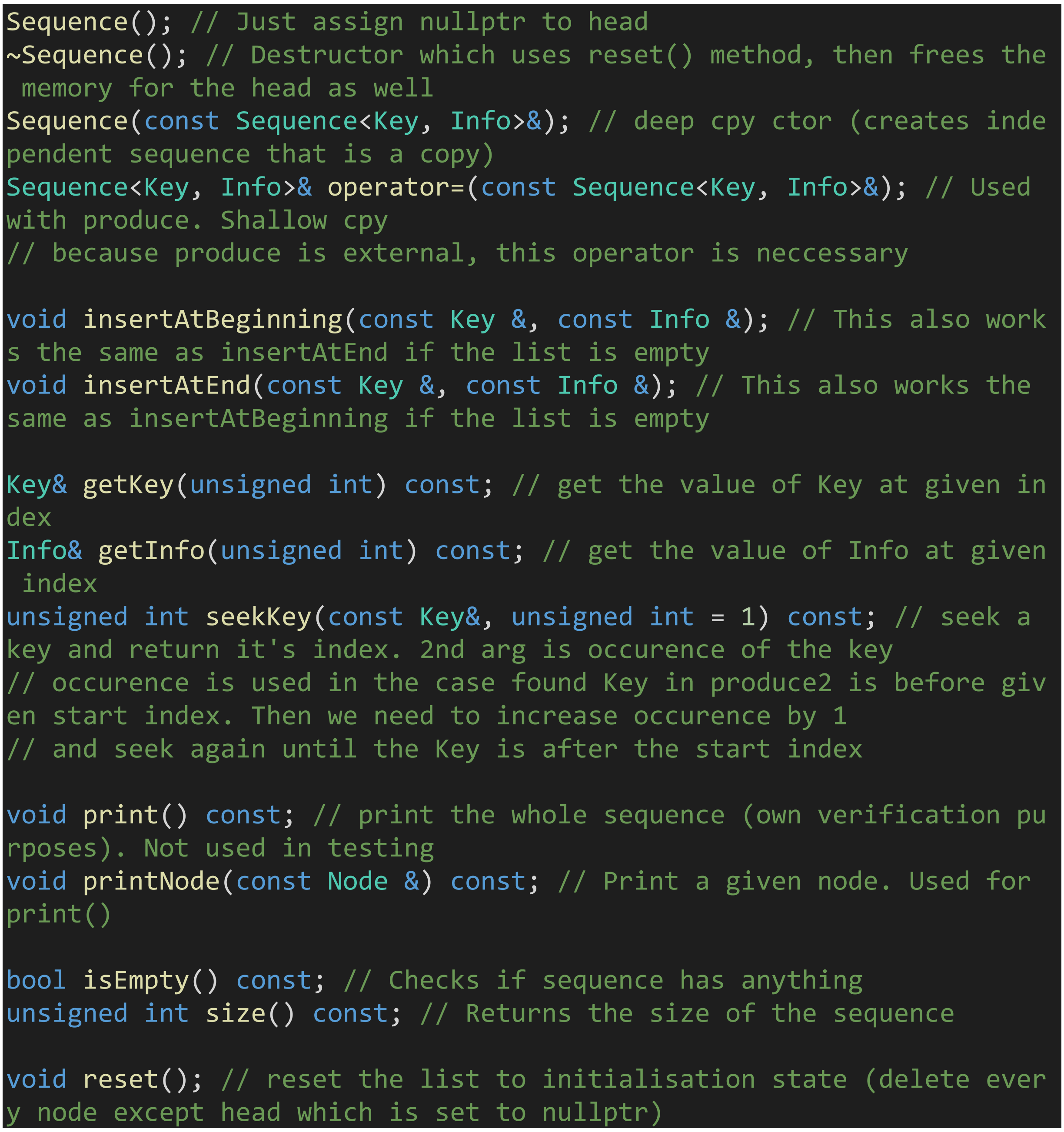
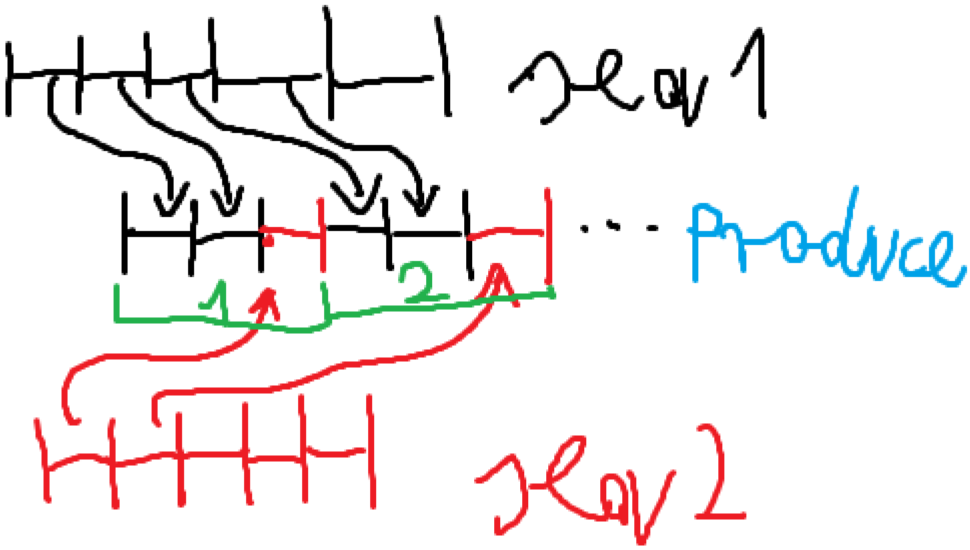
Documentation of the used methods in the `Sequence class`. The same documentation can be found in the header file provided:



```
Sequence(); // Just assign nullptr to head
~Sequence(); // Destructor which uses reset() method, then frees the
 memory for the head as well
Sequence(const Sequence<Key, Info>&); // deep cpy ctor (creates inde
pendent sequence that is a copy)
Sequence<Key, Info>& operator=(const Sequence<Key, Info>&); // Used
with produce. Shallow cpy
// because produce is external, this operator is neccessary

void insertAtBeginning(const Key &, const Info &); // This also work
s the same as insertAtEnd if the list is empty
void insertAtEnd(const Key &, const Info &); // This also works the
same as insertAtBeginning if the list is empty

Key& getKey(unsigned int) const; // get the value of Key at given in
dex
Info& getInfo(unsigned int) const; // get the value of Info at given
 index
unsigned int seekKey(const Key&, unsigned int = 1) const; // seek a
key and return it's index. 2nd arg is occurence of the key
// occurence is used in the case found Key in produce2 is before giv
en start index. Then we need to increase occurence by 1
// and seek again until the Key is after the start index

void print() const; // print the whole sequence (own verification pu
rposes). Not used in testing
void printNode(const Node &) const; // Print a given node. Used for
print()

bool isEmpty() const; // Checks if sequence has anything
unsigned int size() const; // Returns the size of the sequence

void reset(); // reset the list to initialisation state (delete ever
y node except head which is set to nullptr)
```

Here is a graphic that informs us about the way produce method works. In the middle there is a `sequence` that will be returned, `seq1` is at top, `seq2` at the bottom. The chunks are labeled with a green color. Limit parameter tells us how many chunks there should be in the Sequence that's being returned:



`Produce2` has the same way of operating as shown in the above graphic. Only the starting point differs.

I avoided declaring and implementing methods that are not used anywhere in the program. The only ones that are not used are `print()` and `printNode()`, which are – as mentioned in the documentation provided earlier – used only for my own verification purposes during more through testing.