1) The solution code to question 1 is in packages a.bitvector.a and a.bitvector.b

1C) The pattern used for this problem is iterator design pattern. It supports the basic operations that would allow to iterate over any kind of collection.
Few benefits of using this pattern is

- You can implement an Iterator which has the iterating over collection functionality, and we can use it over subclasses and other methods without having to code everytime.
- Objects that need to be traversed over don't need to have their interfaces cluttered up with traversal methods
- You can hand out Iterators to clients you wish, and each client can easily traverse through the collections

---

2) The solution code to question 2 is in packages b.queue

---

3) The solution code to question 3 is in packages c.ast

3B) The pattern used for this problem is adapter design pattern. The adapter pattern allows the interface of an existing class to be used as another interface. Like BitVector is used for NodeSet class.

- It is often used to make existing classes work with others without modifying their source code.
- Helps achieve reusability and flexibility.
- Client class is not complicated by having to use a different interface and can use polymorphism to swap between different implementations of adapters.

---

4) The solution code to question 3 is in packages d.ast

4C) The visitor design pattern provides a method of separating an algorithm on an object and the object's actual class implementation.

- Follows the open/closed principle
- Allows a new operation to be defined without changing the implementation of the class