NAME: _____

```
class BeanJar
{
public:
    BeanJar(int maxBeans, int beans);
    int getBeans() const;
    bool addBeans(int beans);
    bool removeBeans(int beans);

private:
    int maxBeans;
    int beans;
};
```

Figure 1

The declaration of the *BeanJar* class is given in Figure 1. Instances of the *BeanJar* class represent bean jars of varying capacity. The member variable maxBeans represents the capacity of the jar. The member variable beans represents the current number of beans in the jar, which must be less than or equal to maxBeans. The class provides functions to add beans, remove beans, and get the number of beans that are currently in the bean jar. The functions to add and remove beans return a boolean value that indicates whether the operation succeeded or failed. When one of these functions fails, the number of beans in the jar will not have been changed. For example, if a bean jar has a capacity of 10 beans and there are 9 beans currently in the jar, then *addBeans(2)* will return false and the number of beans in the jar will remain at 9.

1) Draw the UML class diagram for the *BeanJar* class given in Figure 1. (25 points)

2) Provide an implementation of the *BeanJar* constructor. (25 points)

3) Provide an implementation of the *getBeans* function. (25 points)

4) Provide an implementation of the *addBeans* function. (25 points)

5) Develop test code to test the *addBeans* function. Make sure the test code provides *good coverage* in the sense that it executes all lines of code in the *addBeans* function. Use the *assert* function in your test code. (25 points)

6) Provide an implementation of the *removeBeans* function. (25 points)

7) Provide an alternative implementation of the *addBeans* function that sets the number of beans in the jar to its maximum when the added beans exceed the available capacity. The function should continue to return false in this case. (25 points)

8) Provide an alternative implementation of the *removeBeans* function that sets the number of beans in the jar to zero when the beans to remove are greater than the number of beans in the jar. The function should continue to return false in this case. (25 points)