# Task 1:

1. Create a class *Exception* which includes one member variable of type *string*. Create a class *ERangeError* which includes two data members: *unsigned*: *m_Max* (to indicate the maximal allowed index) and *m_Used* (to denote the actually used index) both initialized in the constructor. The classes should posses the following interfaces:

```
//The implementation has already been included! Do not change!
class Exception
{
      const std::string m_sDescr;
public:
      const std::string& What() const { return m_sDescr; }
      Exception(const std::string& descr) : m_sDescr(descr) { }
      Exception(const Exception& ma) : m_sDescr(ma.m_sDescr) { }
      virtual ~Exception() { }
};
```

---

```
//You have to implement the constructor and the copy-construtor
class ERangeError : public Exception
{
      const unsigned m_Max;
      const unsigned m_Used;
public:
      ERangeError(const std::string& descr, unsigned max,
                  unsigned used);
      ERangeError(const ERangeError& ma);
      virtual ~ERangeError() {}

      unsigned Max() const { return m_Max; }
      unsigned Used() const { return m_Used; }
};
```

2. Construct a class *CBooleanVector* which represents a Boolean vector. The class *CBooleanVector* must incorporate a nested friend class *CProxy* (which includes a reference to an element of *CBooleanVector*). Implement all the functions that are declared in the following code:

```
class CBooleanVector
{
      unsigned m_nElements;   unsigned* m_arElements;

      class CProxy;
      friend class CProxy;
      class CProxy
      {
            unsigned& m_VectorElement;
      public:
            CProxy(unsigned& VectorElement);
            CProxy& operator= (unsigned r);  //lvalue
            operator unsigned() const;       //rvalue
      };
```

```
public:

      CBooleanVector(unsigned el = 0);
      CBooleanVector(const CBooleanVector& v);
      virtual ~CBooleanVector();

      const CProxy operator[](unsigned pos) const;
      CProxy operator[](unsigned pos);

      CBooleanVector& operator=(const CBooleanVector& rv);
      friend std::ostream& operator << (std::ostream& os,
                                        const CBooleanVector& v);
};
```

3. Provide for exception handling. All the functions that manage indices should generate *ERangeError* exceptions (in case if an index range is exceeded). An exception *Exception* is generated whenever a non Boolean value is assigned to an element of the vector.

4. Test your classes with the following *main* function:

```
int main()
{
      using namespace std;
      try
      {
            CBooleanVector v1(3);

            //v1[0] = 5; //test
            //v1[3] = 1; //test
            //v1[-5] = 0; //test

            v1[1] = 1;
            cout << v1 << endl;
            CBooleanVector v2;
            cout << (v2 = v1) << endl;
            CBooleanVector v3 = v2;
            v3[0] = 1;
      }
      //catch all your exceptions here and
      //display all the available information about them.

      return 0;
}
```

5. Try to uncomment each of the marked lines and check how the exceptions are generated and processed.