1. A string tokeniser is an object that is passed a line of text as a parameter at the time it is created. The tokeniser breaks up the line into its component tokens (usually words), and returns the tokens one by one. Write a string tokeniser class that performs these tasks. It should provide three methods. The first should be a constructor, which accepts the string to tokenise as a parameter, and another string containing one or more delimiter characters. At some point, the first string must be broken into a sequence of tokens, each separated by at least one delimiter character. Your class should not impose any limits on the length of the string to be tokenised, and it should leave the original string intact and unmodified.

The second method, \texttt{getNext} should return, as a string, the next token in the original string that has not yet been returned. Note that you will need to allocate the memory for these returned strings dynamically. There should be no maximum length of the token that can be returned. Finally, the method \texttt{hasMoreTokens} should return true if there are more tokens that have not yet been returned, false otherwise.
2. Write a C++ class to represent a stack of integers. Using this stack class, write a program that reads in simple integer expressions in postfix notation, evaluates the expression and writes out the result. The input will consist only of integer numbers and the operators +, -, *, /. For example, if the input is:

```
4 3 2 + 8 * +
```

the output would be: 44

You can assume that there will always be a space between each number and/or operator, and that the input is always a valid postfix expression. You can also assume that there already exists a string tokeniser class (see question 1) that can be used to divide the original string into substrings separated by spaces.

3. A model of a three-dimensional object may be represented as a three-dimensional volume of equal sized cubes called "voxels". This can be represented as a three-dimensional array of 1's and 0's, where a 1 represents the presence of part of the object in that position, and a 0 the absence.

e.g. (in 2D):

```
\begin{array}{ll}
\text{can be modelled as}: & \begin{array}{|c|c|c|c|}
\hline
1 & 1 & 0 & 1 \\
\hline
1 & 0 & 1 & 0 \\
\hline
0 & 1 & 0 & 0 \\
\hline
0 & 0 & 0 & 1 \\
\hline
\end{array} \\
\text{and be represented by}: & \{ [0,0,1,1,0,0], [0,0,1,1,0,0], [1,1,1,1,1,1], [0,0,1,1,0,0], [0,1,0,0,1,0], [1,0,0,0,0,1] \}
\end{array}
```

The class definition of such an object should also give the location of its centre in Cartesian space (i.e. integer x, y and z co-ordinates), where each voxel is a one unit cube of Cartesian space. Write a C++ program which implements such objects, and detects collisions between them i.e. tests two or more of such objects at different locations to see if they are intersecting. (You may assume that the objects are not rotated).
4. Write a linked list class to represent lists of items. Your class should be a template class, so that the type of object to be stored is specified at the point where the list is declared. Your linked list class should have an associated iterator class, that can be used to step through the items in the list. Using an iterator, it should be possible to insert, delete and modify list items at the list element where the iterator points. Your list class and iterator class should both provide a constructor and destructor.

5. What is the standard template library (STL)? Briefly describe the most important container classes in the STL. With many class libraries of container data structures, it is necessary to have a separate version of the code for each combination of data structure, type of contained object, and algorithm that can be applied. Explain how the STL avoids this sort of code duplication using templates and iterators. (14 marks)

   The main iterator classes in the STL provide the `begin()`, `end()` and `++` operations. Show how a programmer implementing the STL might implement these operations if, internally, the container uses (i) an array, (ii) a linked list. (6 marks)

6. Write a class to represent rational numbers. Your class should provide +, −, * and / operators, and should also overload the << and >> operators for input and output. Write a small template method that takes two numbers as parameters, and writes the sum of the numbers to the screen. The method should work with both integers and rational numbers as the template type.

7. Most Windows programs are event-driven. Describe the main differences between event-driven and traditional programming. Describe how external events from input devices lead to messages being received by particular programs or windows, with particular reference to message queues. (6 marks)

   Write an MFC program that displays a blank window. When the left mouse button is clicked, a small X shape should appear at that point in the screen. If there are any existing X’s on the screen, a line should be drawn to connect the newly placed X to the nearest existing one. To do this, you will need to keep a list of the locations of existing clicked points on the screen. Your program should repaint the window whenever it receives a WM_PAINT message. (14 marks)