a. In C++ you can easily write a function that will work on any one dimensional array of a particular type, regardless of its size, so long as you tell it what the size is. For example

```
double average_one_dimension(int A[], int N)
{ int total = 0;
  for (int i = 0; i < N; i += 1)
    total += A[i];
  return (double) total / N; }
```

but it doesn't work that way for two dimensions. For example, this won't work

```
double average_two_dimensions(int A[][], int M, int N)
{ int total = 0;
  for (int i = 0; i < M; i += 1)
    for (int j = 0; j < N; j += 1)
    total += A[i][j];
  return (double) total / (M * N); }
```

Explain why.
This does not need to be a long answer, but it must really explain.

b. Define a class, using public and protected in an appropriate way, to implement a three dimensional array of ints. It must be able to be used effectively in these ways (or a close approximation if you can't quite get the detail):

```
void f()
{ threeDints A(7, 41, 12); // should be similar to int A[7][41][12];
  A.at(3, 33, 9) = 66;     // should be similar to A[3][33][9] = 66;
  cout << A.at(i, j, 3);   // should be similar to cout << A[i][j][3];
  A.at(1, 1, 1) *= 2; }    // should be similar to A[1][1][1] *= 2;
```

Four example uses does not necessarily mean that you need four functions or methods.