Object file examination

1. Compile the given files into a binary program
2. Find the symbol names for the add and average functions that are defined in the object file for templates.cpp:

   ```
   058 00000000 SECT9    notype ()       External         | ?add@@YANNN@Z
   (double __cdecl add(double,double))

   178 00000000 SECT58 notype ()        External
   | ?average@@YANV?$vector@NV?$allocator@N@std@@@std@@@Z
   (double __cdecl average(class std::vector<double,class std::allocator<double> >))
   ```

Function templates

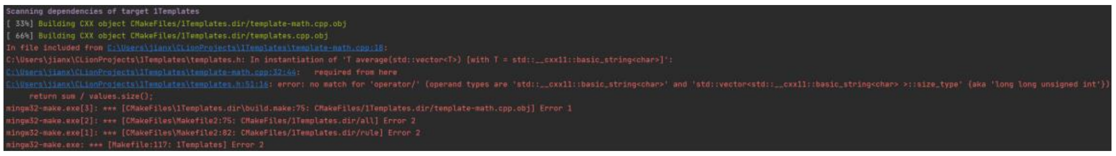
1. Convert the add() and average() functions to function templates. Compile the project again — what symbols do you see in templates.o[bj] now?
   Nothing.

2. Try to call average() with the types int, double and string from the main() function in template-math.cpp. What errors do you see? Why? Fix what you can (and remove what is fundamentally impossible).
   Apparently, string type can't be calculated average. So it is removed.
   When call average(), it said undefined reference. We have to move t he implementation of average() to templates.h. Then it worked well with types int and double.

3. What result does average({ 1, 2, 3, 4 }) give? Why? Modify this call to explicitly pass a double type to the template parameter and check that it now returns the correct value.
   It gives 2. Because both the denominator and numerator are int type. We can specify double type enclosed in angle brackets when call average().

4. Move the implementation of add() and average() to templates.h. Now make everything compile and run again. This is a common pattern with templates (including the standard template library): you need to put template implementations in header files so that the compiler can instantiate new versions of functions and classes when compiling code that uses the templates. Try explaining this to your neighbour.
   No neighbor. Explained to myself.

5. Modify main() to call average() with the types int, double and string. What errors do you see? Why?
   When call average() with the type string, There are build errors:



```
Scanning dependencies of target 1Templates
[ 33%] Building CXX object CMakeFiles/1Templates.dir/template-math.cpp.obj
[ 66%] Building CXX object CMakeFiles/1Templates.dir/templates.cpp.obj
In file included from C:\Users\jianx\CLionProjects\1Templates\template-math.cpp:18:
C:\Users\jianx\CLionProjects\1Templates\templates.h: In instantiation of 'T average(std::vector<T>) [with T = std::__cxx11::basic_string<char>]':
C:\Users\jianx\CLionProjects\1Templates\template-math.cpp:32:44:   required from here
C:\Users\jianx\CLionProjects\1Templates\templates.h:51:16: error: no match for 'operator/' (operand types are 'std::__cxx11::basic_string<char>' and 'std::vector<std::__cxx11::basic_string<char> >::size_type' {aka 'long long unsigned int'})
     return sum / values.size();
mingw32-make.exe[3]: *** [CMakeFiles\1Templates.dir\build.make:75: CMakeFiles/1Templates.dir/template-math.cpp.obj] Error 1
mingw32-make.exe[2]: *** [CMakeFiles\Makefile2:75: CMakeFiles/1Templates.dir/all] Error 2
mingw32-make.exe[1]: *** [CMakeFiles\Makefile2:82: CMakeFiles/1Templates.dir/rule] Error 2
mingw32-make.exe: *** [Makefile:117: 1Templates] Error 2
```

Because string can not be divided by int.

Class templates

1. Convert class GrowableArray into a template that takes a template parameter typename T instead of assuming it will always hold double values. Is there any reason why the new GrowableArray template couldn't hold non-numeric values like std::string values?
   I don't understand. The new GrowableArray template can hold std::string value.

2. Instantiate a couple of GrowableArray objects in main() with different types, e.g., GrowableArray<int>, GrowableArray<string> and GrowableArray<vector<GrowableArray<double>>>. Add values to them and check that they work.
   The last one doesn't work. Because GrowableArray<double> value can not be added into vector.

Extra (optional) exercises

1. Modify the GrowableArray template to take another parameter: the size of the smallArray_ field below which the in-object-storage optimiation should be applied.
   Please see "templates.h" line 59, 64.

2. Add an average() method to the GrowableArray template. Comment on how well it works with GrowableArray<int>, GrowableArray<double> and GrowableArray<string>. Can you instantiate a GrowableArray<string> if you don't call the average() method? Why / why not?
   Please see "templates.h" line 113-137. Adapted from average(). When it's int, the average output is also int. It works well with double type. It can not be built when its string type.
   Yes. The class won't call its member functions until those functions are explicitly called.