# Lab Report (Lab 04)

ID: 102946740

Name: Benjamin Fekete

Q.1.1 What do the < and > mean or indicate?

< and > is declaring parameters for a static array, so in this example (array<int, 3>) we are setting the data type to integer with a size of 3 elements.

Q.1.2 Why don't we need to write std:array here? (Is this good?)

We don’t use std:: because we are using the namespace std, this generally isn’t good practice as you may have conflicts with other libraries using similar functions, data types, structs etc. so its best practice to manually use std:: as if there is another library included, you may have conflicts with similar or same named items.

Q.1.3 Explain what the int and 3 indicate in this case?

As said above, it sets the data type of the array to an integer, then the size to 3 elements.

Q.1.4 In the code above, what is the type of itr2?

Using the auto keyword, detects the type of a1.Begin (whatever your trying to set the auto variable to) and assigns it that data type, in this case since a1 is set to an integer array it, itr2 will be set to an integer array as well.

Q.1.5 In the code above, what is the type of v?

V is set to be a pointer to integer or int&

Q.1.6 In the code above, what does the & mean in (auto& v : a1)

It means that the auto type will assign the detected data type but also set it to be a pointer to that type. So in this case a pointer to type int.

Q.1.7 Try this. Why does a1[3] work but at(3) does not?

In this example, we are trying to do something with the 3rd element, however the AT method does not worth the same as [3] way of retrieving data from an array, instead it tries to find the first MATCHING element with the same value as the input, so AT(3) is looking for the first instance of int 3, if it finds 3 it will return the location of the array element, so if 3 was the 3rd element it would return 2. But since the array doesn’t have a int 3, it will return out of bounds.

Q.1.8 How would you do a forward (not reverse) sort?

Q.2 In array\_demo\_2, explain what a4(a1) does

This is a constructor statement, its using a constructor method to assign the value, in this case its setting a4 to the value of a1.

Q.4 How do we (what methods) add and remove items to a stack?

The pop() method removes the value from the stack (but does not return as a value)

Q.5 A stack has no no [] or at() method - why?

Stack makes use of last in first out (LIFO), meaning what ever was last placed into the stack, it’s the first to be used/removed. This removes the use of operators like [] and () as its not part of its core use.

Q.6 What is the difference between a stack.pop() and a queue.pop()

Stack.pop removes the element at the top of the queue, where queue,pop removes the last added element.

Q.7 Can we access a list value using and int index? Explain.

Not directly, std::list doesn’t support any accessing operators like [] or at().

Q.8 Is there a reason to use a list instead of a vector?

Yes, lists are efficient at inserting and deleting data, have more stability with iterations (for loops), lists also don’t reallocate memory avoiding overhead.

Q.9 Was max\_size and size the same? (Can they be different?)

Max size returns the theoretical maximum amount of elements (set by the system or library) and size() returns the current amount of elements in the vector.

Q.10 Which ParticleClass constructor was called?

Q.11 Were the ParticleClass instances deleted? If so, how?

ParticleClass was all destructed (confirmed in output) because the objects were out of scope, meaning once the function was completed all local variables were destructed.

Q.12 Was the vector instance deleted? If so, how do you know this?

Yes, by checking the debugging menu

Q.13 Your IDE might suggest to use emplace\_back instead of push\_back. What does this mean?

This is a suggestion for oop, making a more efficient object construction.

Q 14 Apparently const prevents a copy - quicker performance. Is this true? Use evidence or sources to justify your answer.

Const does not prevent copying, it just prevents modification of an object.