

Forum: Student Discussions **Collection**

# Collection

Users can Collect posts into a printable, sortable format. Collections are a good way to organise posts for quick reading. A Collection must be created to tag posts

[More Help](#)

Print Preview

Filter

Sort by

Date of Last Post

Order

▼ Descending

Select: [All](#) [None](#)

Mark

▼

- Thread:

Assessment 1?

Post:

RE: Assessment 1?

Author:

Alen Alempijevic

Posted Date:

28 March 2017 9:11 AM

Status:

Published

Done, available in same folder as the Assessment 1 Description

Reply

Quote

Mark as Read

Thread:

Assignment 1 - sensor allocation list

Post:

RE: Assignment 1 - sensor allocation list

Author:

Alen Alempijevic

Posted Date:

28 March 2017 9:07 AM

Status:

Published

Nicolas,  
  
Sensor allocation has been uploaded to same folder as the Assessment Description

Reply

Quote

Mark as Read

Thread:

Posted Date:

27 March 2017 9:23 PM

https://online.uts.edu.au/webapps/discussionboard/do/message?course\_id=\_30450\_1&nav=discussion\_board\_entry&requestTyp... 1/5

Assignment 1 - sensor allocation list

**Status:**

Published

**Post:**

Assignment 1 - sensor allocation list

**Author:****Nicolas Giovanangeli**

Hi everyone,

Does anyone know where to find the list of which sensor we are allocated to for assignment 1? I haven't been able to find anything and I need to get an early start on this thing.

Cheers,  
Nicolas

[Reply](#)[Quote](#)[Mark as Read](#)**Thread:** Assessment 1?**Posted Date:** 27 March 2017 6:44 PM**Post:** RE: Assessment 1?**Status:** Published**Author:****Navi Gunaratne**

Thank you Alen,

Could you please upload the sensor allocations?

[Reply](#)[Quote](#)[Mark as Read](#)**Thread:** Assessment 1?**Posted Date:** 27 March 2017 5:29 PM**Post:** RE: Assessment 1?**Status:** Published**Author:****Alen Alempijevic**

Assessment 1 has been online from this morning, under the Assessments tab

[Reply](#)[Quote](#)[Mark as Read](#)**Thread:** C++ Version Issue**Posted Date:** 27 March 2017 5:29 PM**Post:** RE: C++ Version Issue**Status:** Published**Author:****Alen Alempijevic**

For C++11 features you need to let the compiler know via CMake, add below to your CMakeLists.txt

```
set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -std=c++11")
```

[Reply](#)[Quote](#)[Mark as Read](#)

**Thread:** Assessment 1?

**Posted Date:** 26 March 2017 3:14 PM

**Post:** RE: Assessment 1?

**Status:** Published

**Author:**  **Navi Gunaratne**

Hi,

I am having the same issue, have not been able find any information regarding Assessment 1 beyond the submission link.

Could someone update?

Thanks,


[Reply](#)[Quote](#)[Mark as Read](#)

**Thread:** C++ Version Issue

**Posted Date:** 26 March 2017 6:20 AM

**Post:** C++ Version Issue

**Edited Date:** 26 March 2017 6:22 AM

**Author:**  **Jian Zhan**

**Status:** Published

Hi,

I am using the Ubuntu provided on UTS Online but when I tried to use the range-based "for" loop it gave the error saying the function is not supported in c++89. But somehow I still can use the rand() function, which I guess it is a function can only be used in c++11 or beyond. Could someone tell me what is happening? Thank you!

The screenshot shows the Qt Creator IDE with a C++ file named `working.cpp`. The code contains a range-based for loop, which is not allowed in C++98 mode. The error message in the Issues panel states: "range-based 'for' loops are not allowed in C++98 mode". The code is as follows:

```

83
84
85     cout << s[0]<<endl;
86     cout << s[1]<<endl;
87
88     */
89
90
91     int s[]={1,2,3,4,5};
92     for (int i :s)
93     {
94         cout << " " << i;
95     }
96     return 0;
97
98
99 }

```

The Issues panel at the bottom shows the error details:

```

In function 'int main(int, char**)':
range-based 'for' loops are not allowed in C++98 mode
[CMakeFiles/working.dir/working.cpp.o] Error 1

```

Yours Faithfully,

Jian Zhan

Reply

Quote

Mark as Read

**Thread:** General Code Debugging **Posted Date:** 9 June 2016 2:44 PM  
**Post:** General Code Debugging **Edited Date:** 9 June 2016 2:47 PM  
**Author:** **Status:** Published



**Alen Alempijevic**

Some students have contacted me for help to debug their code. Debugging code is an important skill to learn as part of programming for mechatronic systems and if we were to give you very specific help to debug your code you would learn far less from this subject. Also the person best equipped to debug your code IS YOU. You know how its designed, why it is that way, and what you expect it to do. To help you debug your code here is some general advice which I gave to students who contacted me.

When it comes to debugging code there are a couple of options. The simplest is to identify potential sources of errors and systematically remove them from the code and/or add debugging outputs (printfs or couts with values of critical variables) until you identify the source of your problem. Then build it back up piece at a time, compiling and running it each time, trying to resolve each bug as it is introduced. Unfortunately the more complex your code gets the harder it is to remove pieces without affecting other parts of the code.

Another option when you have code that terminates unexpectedly (e.g.. SEG FAULT) is to use a debugger such as GDB to examine the state of your program at the time of the crash. You should be able to find examples of using gdb to debug a crash online. If the code is exiting without any error message it may not be crashing but simply exiting unexpectedly.

Debugging can be frustrating and can cause you to question your understanding of core concepts. In cases like this I recommend constructing simple examples for yourself, separate to your project, to confirm some of the core concepts you are working with, such as mutexes. Mimic the basic application of the concept in your project but without the complexity of the other requirements. Once you are sure you understand the concept reexamine the way you use it in your code and if necessary strip it back to be as similar to your simple example as possible and build up from there. Generally speaking the key to building bug free code is to build and test it often so you can catch bugs AS THEY ARE INTRODUCED. This gives you a greater deal of certainty about which lines of code are at fault.

If you are battling with this type of debugging you may also be starting to appreciate the need for having well contained classes/objects, avoiding globals and writing unit tests.

[Reply](#)[Quote](#)[Mark as Read](#)

Select: [All](#) [None](#)

[Mark](#)[← OK](#)