Assignment 4

SWARNARUP BHUNIA

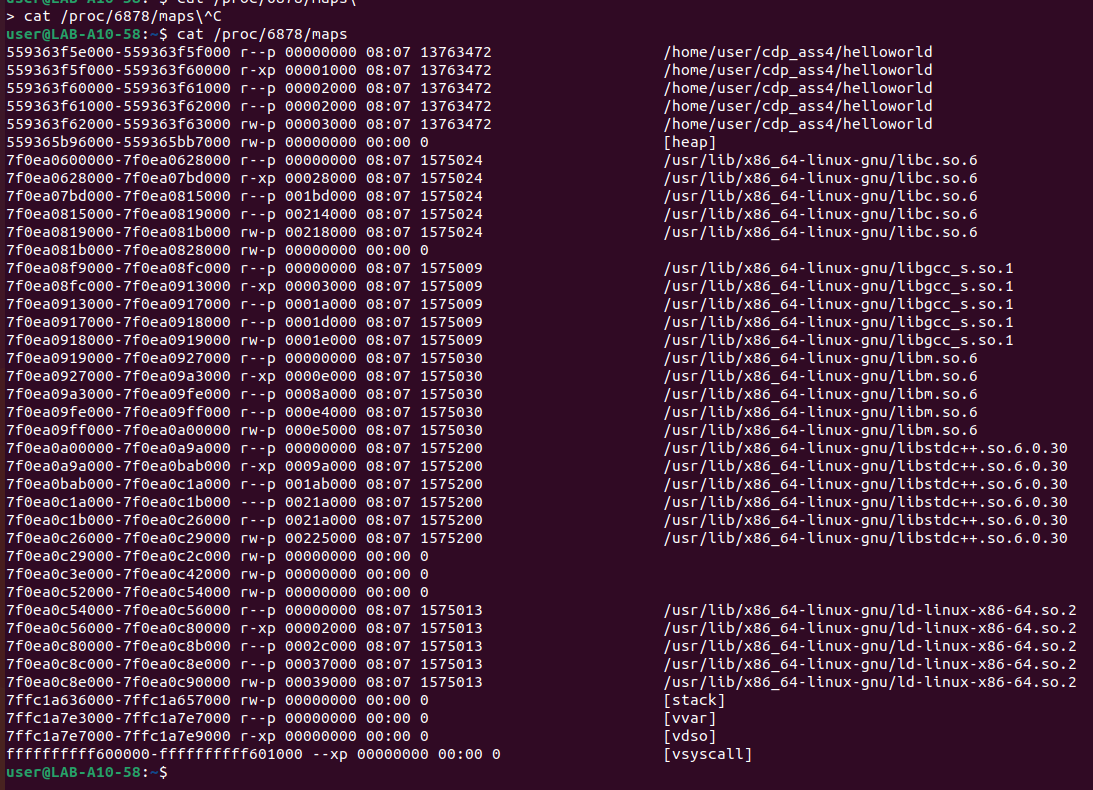
B21140

Q1.



<address start>-<address end> <mode> <offset> <major id:minor id> <inode id> <file path>

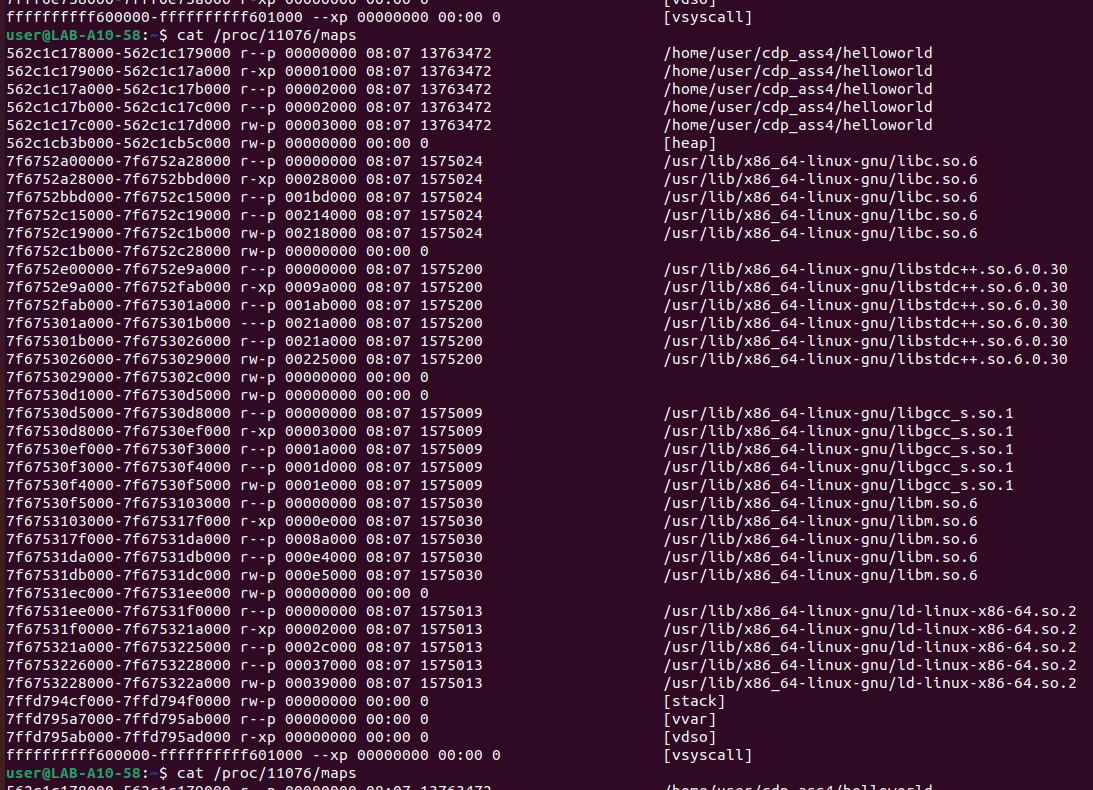
559b8c418000-559b8c41a000 r--p 00000000 08:30 1708 /usr/bin/cat

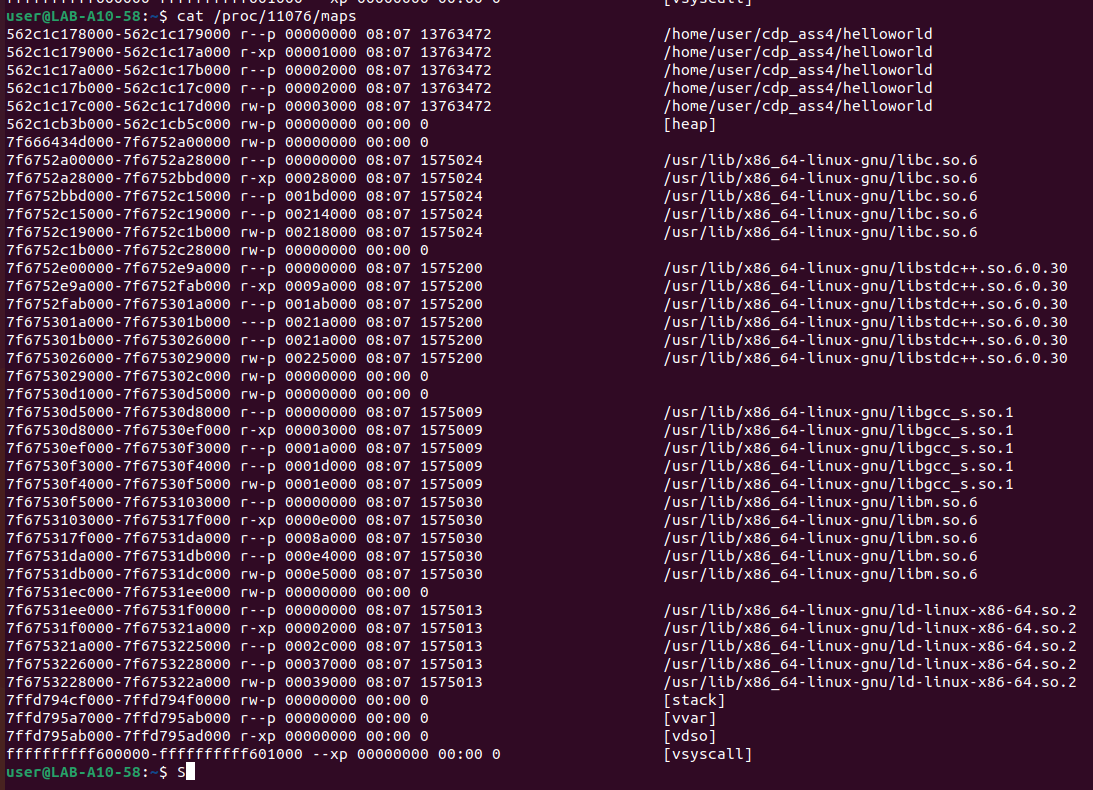


* **address start** – address end is the start and end address of that mapping. Note that the whole output is sorted based on those addresses, from low to high.
* **mode** (permissions) specifies which actions are available on this mapping and if it’s private or shared.
* **offset** is the start offset in bytes within the file that is mapped. This only makes sense for file mappings. For instance, stack or heap mappings are examples of mappings that aren’t files, and in those cases, the offset is 0. In the above example, the mapping is of the */usr/bin/cat*file, and the offset is 0.
* **major:minor ids** represent the device that the mapped file lives in the form of a major and minor id. In the above example, 08:30 represents the major and minor id of the hard drive that has the root filesystem. For non-file mappings, this column shows 00:00*.*
* **inode id** of the mapped file (again, that’s only valid for file mappings). Inodes are data structures that contain the core filesystem-related metadata. When it comes to non-file mappings, this field is set to 0. In our example, this id is 1708.
* **file path** of the file for that mapping. In the event that this is not a file mapping, that field is empty.

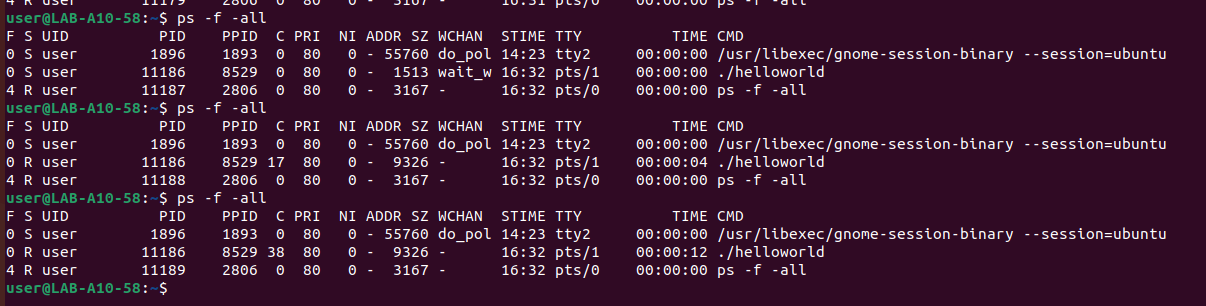
Q2.

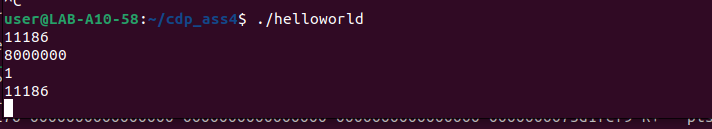
a)



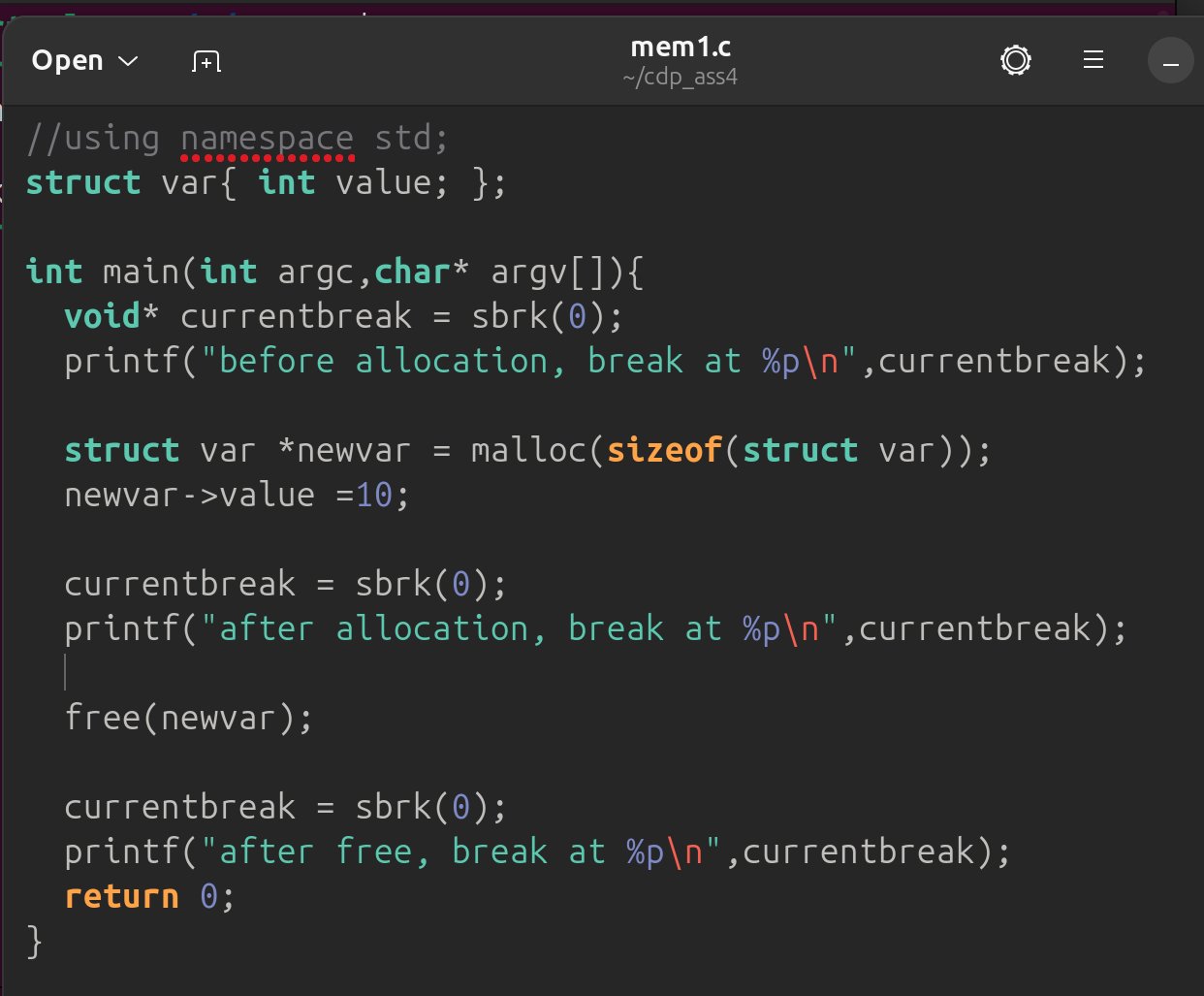


b)



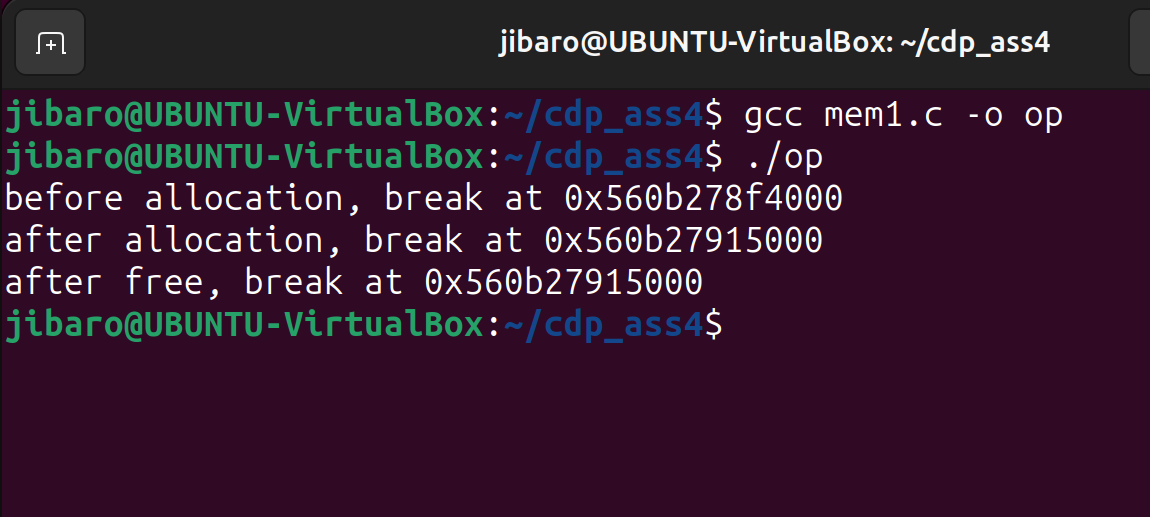


Q3.

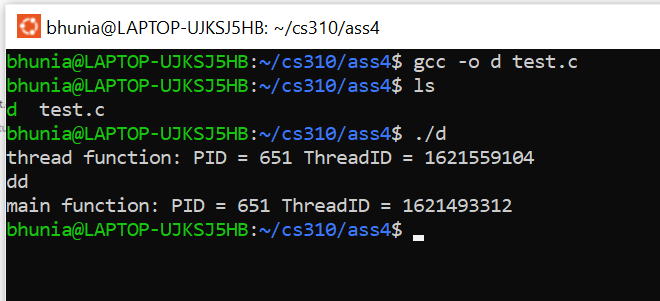
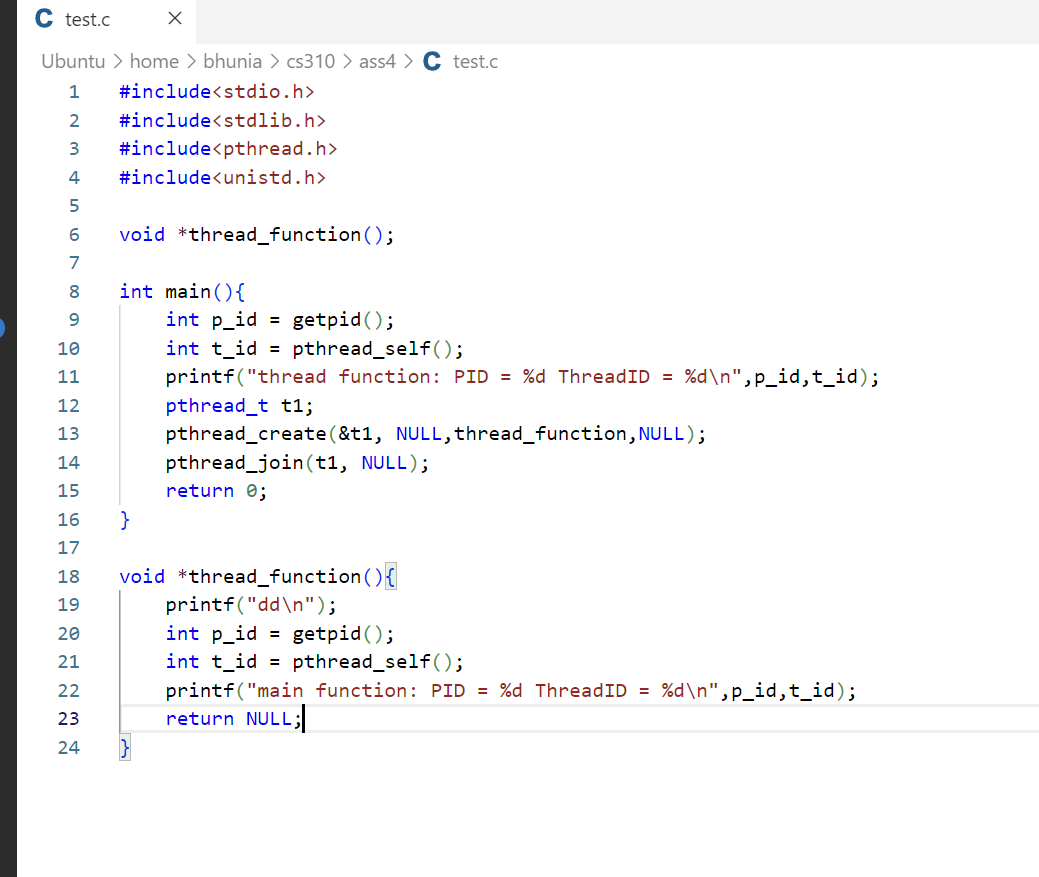


# after allocating memory the programme break increased

#after free programme break did not revert back.

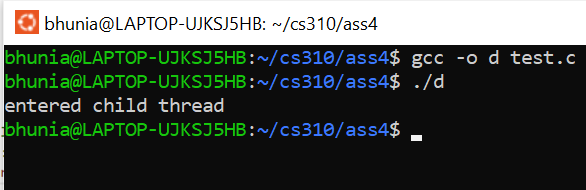
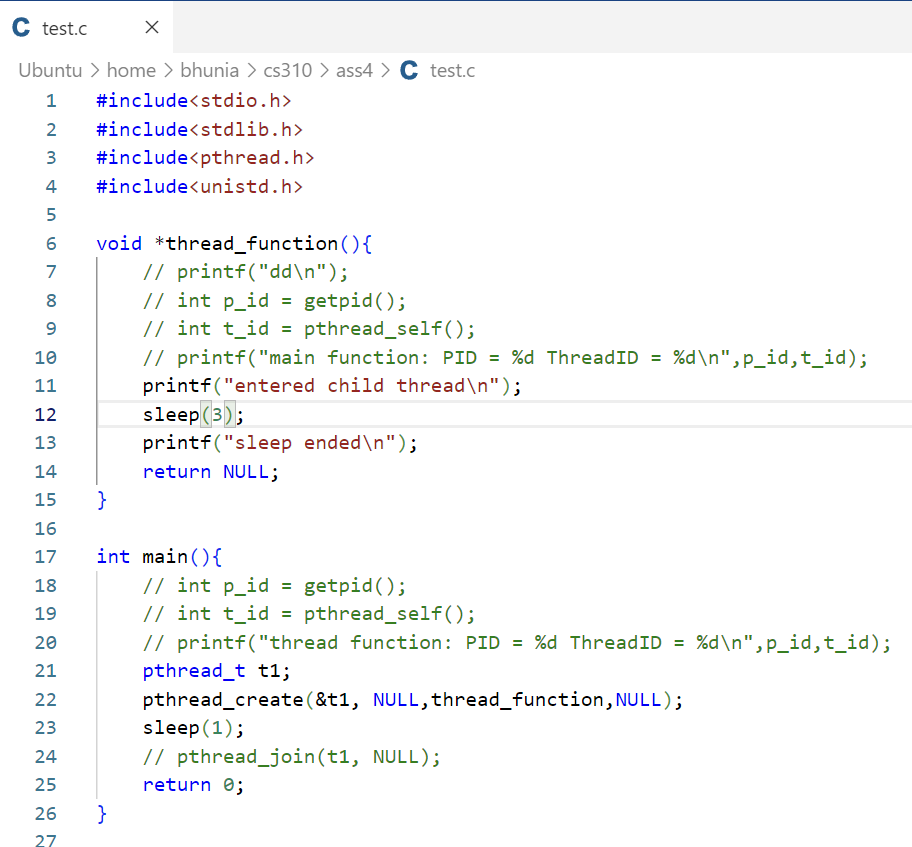


Q4.

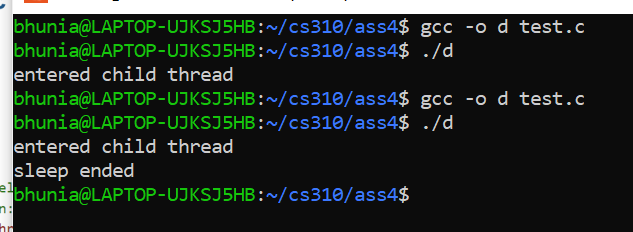
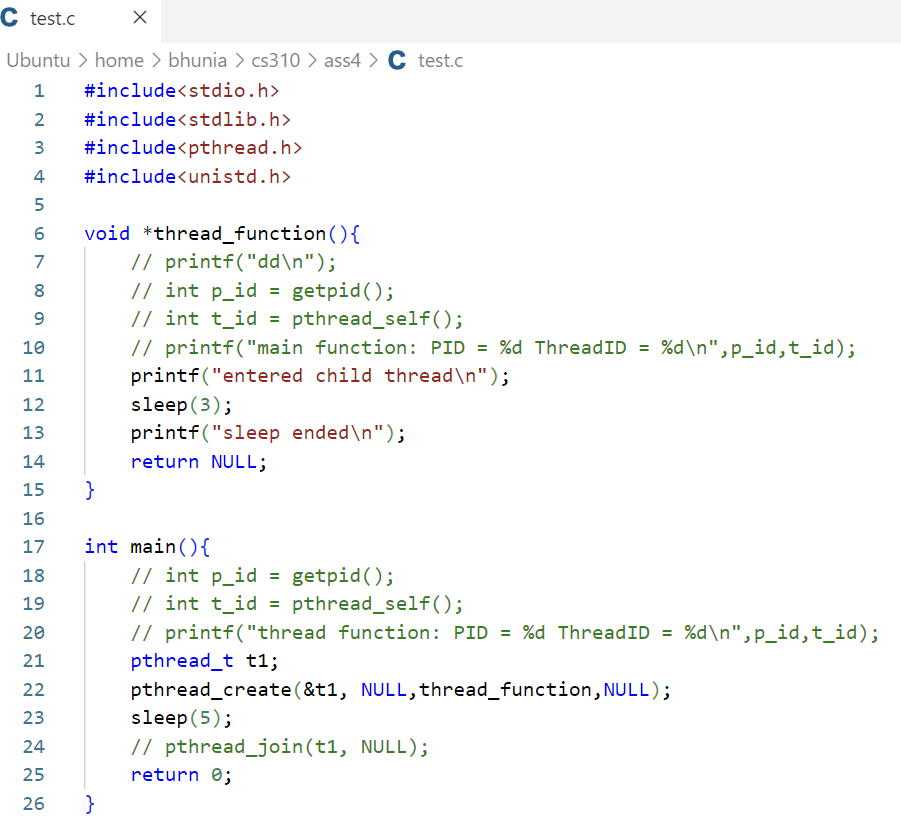
 Threadid is different but pid is same.

a)

case1: thread waits 3 second; main waits 1 second;



Case2: thread waits 3 second; main waits 5 seconds;



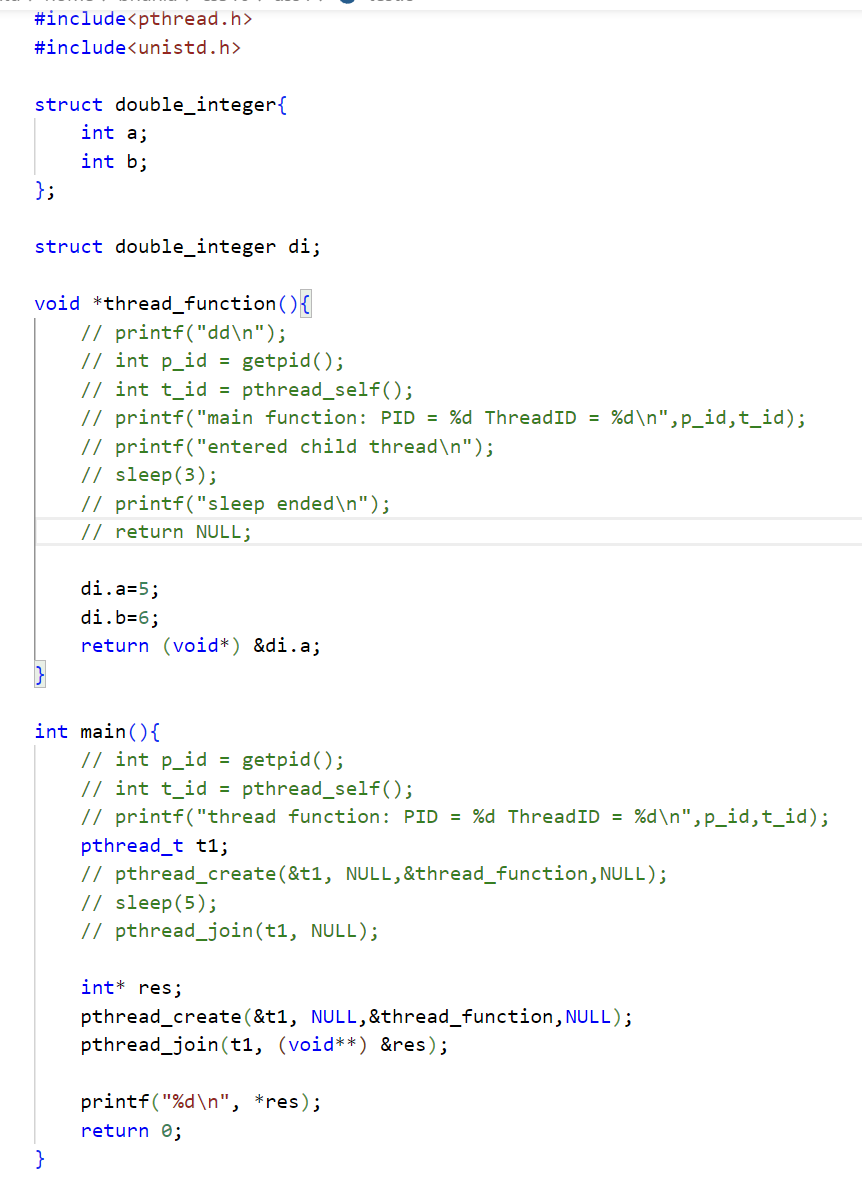
Hence it is clear that main does not wait by default for the thread to finish it’s job. That’s why in case1 printf(“sleep ended”) did not get executed

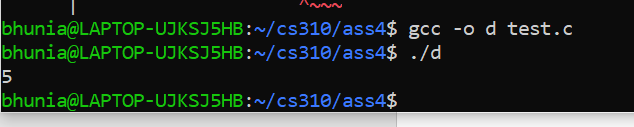
b)

Q5.

a)

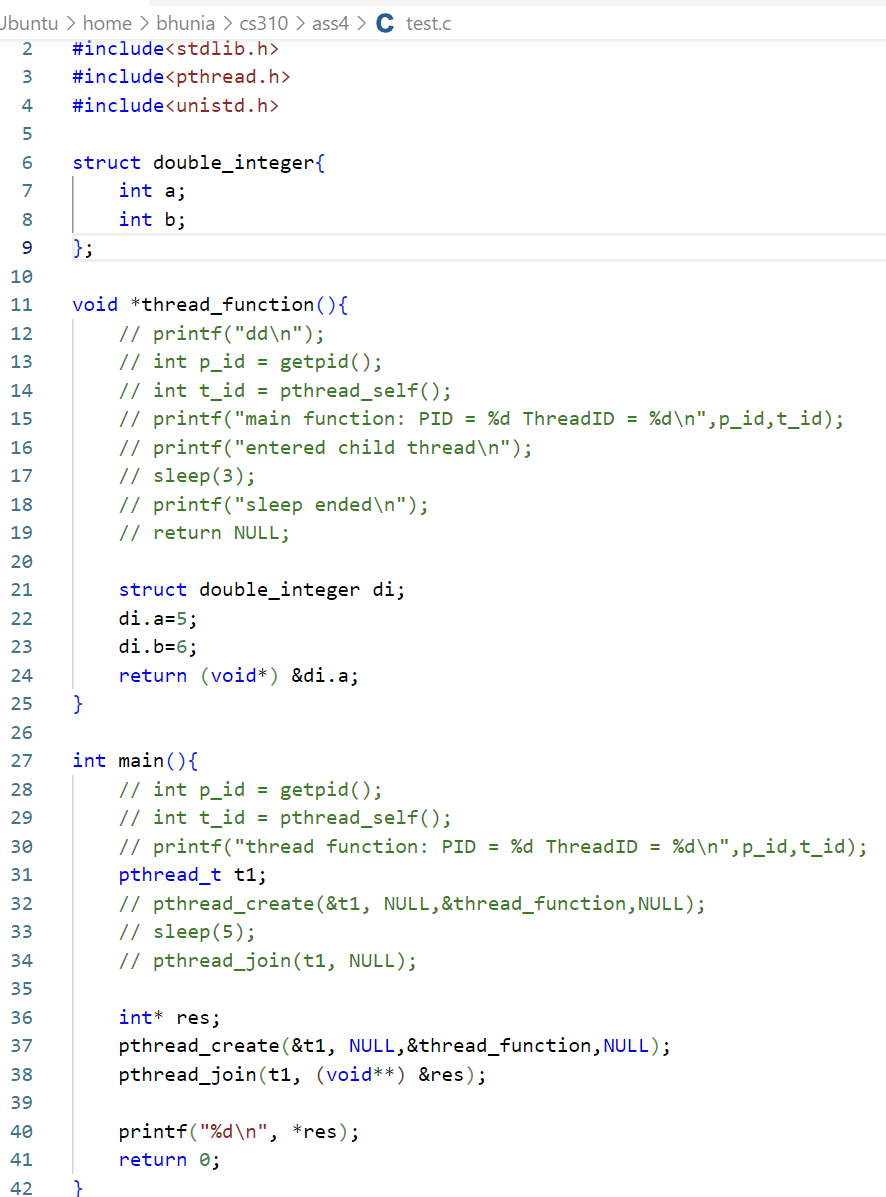
Case 1: The variable di is instantiated outside thread;

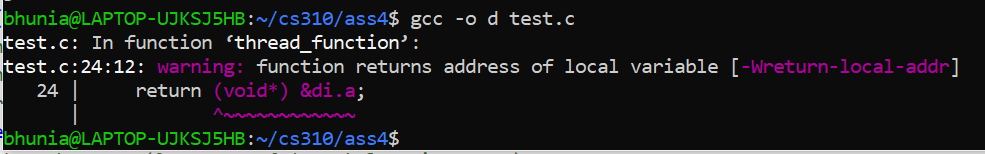




No error in compilation;

Case 2: The variable di is instantiated inside the thread;





It throws compilation error; the item is in the stack of the thread\_function thread; it cannot be called to main function;

b)

**the fix** for this is to take **another reference to an integer with malloc** and return that pointer to main function.

