Inlab 6 - Doxygen & Sphinx

Due: 23.59, 21 September

This Inlab is based on Doxygen and Sphinx. Check the resources to learn more about them before starting to work. Docker does not have Sphinx installed, so you must install and use it on your system. Inlab resource files can be found here.

Follow the naming convention of the submission directory strictly. The structure is given at the end for ease of reference. Q2 and Q3 will be auto-graded. Q1 PDF/HTML will be manually checked.

IMPORTANT FOR AUTOGRADING: Follow the commenting format given in this link for Q2 and Q3.

Utilities

You can use the scripts to clean the build files and start fresh for any part. This is important if you want to avoid repeating the configuration steps of Sphinx for any questions.

Q1. DSA Library (Doxygen & C++) (25)

In this question, you need to create documentation for a data structures library (file DSA.cpp has been provided in inlab06res/q1) by adding detailed comments to all classes and functions. The tester.cpp file is only for your reference on library usage.

Note: Visual Studio Code has auto-documentation support for Doxygen & Sphinx.
The steps (and rubrics) are given below:

(a) File header (2.5)
Add a suitable Doxygen file header that describes what the code is about. The header should contain the file, author, date, and a brief description. You can include extra fields as well.

(b) Class descriptions (2.5)
Add class descriptions for each of the data structures.

The description should include the utility/member functions of that data structure.

(c) Function descriptions (15)

1. Constructors (5)
   Add documentation for all types of constructors of each data structure class.
   It should contain a brief description, input and output parameters (if any), and a list of available functions.
   Add one-line descriptions for each of the variables of the class as well.
2. Printers (2.5)
   Add documentation to all the printing functions.
   It should contain a brief description, input parameters, output parameters, and return details (if any).
3. Member Functions (7.5)
   Add documentation to all the member functions (insert, find, delete, print, reverse, etc.).
   It should contain a brief description, input parameters, output parameters, and return details (if any).

(d) PDF (5)

Zip the latex folder generated by Doxygen and upload it to Overleaf as a new project.
Compile the file **refman.tex** and download the PDF generated.
Add it to submission (q1.pdf) as indicated in the submission directory structure.

Q2. Sphinx Basics (Sphinx & Python) (5)

IMPORTANT FOR AUTOGRADING: Follow the commenting format given in this link for Q2 and Q3.

You have to document the binary search function (given in inlab06res/q2).

Note: Visual Studio Code has auto-documentation support for Doxygen & Sphinx.

The steps (and rubrics) are given below:

(a) Configuration for sphinx-quickstart (2)

Separate source and build directories (y/n) [n]: y
Name prefix for templates and static dir [_]: .
Project name: inlabq2
Author name(s): <roll-number>
Project release []: (leave as is, press enter/return)
Project language [en]: (leave as is, press enter/return)
Source file suffix [.rst]: (leave as is, press enter/return)
Name of your master document (without suffix) [index]: (leave as is, press enter/return)
autodoc: automatically insert docstrings from modules (y/n) [n]: y
doctest: automatically test code snippets in doctest blocks (y/n) [n]: y
intersphinx: link between Sphinx documentation of different projects (y/n) [n]: (leave as is, press enter/return)
todo: write "todo" entries that can be shown or hidden on build (y/n) [n]: (leave as is, press enter/return)
coverage: checks for documentation coverage (y/n) [n]: y
imgmath: include math, rendered as PNG or SVG images (y/n) [n]: y
mathjax: include math, rendered in the browser by MathJax (y/n) [n]: y
ifconfig: conditional inclusion of content based on config values (y/n) [n]: y
viewcode: include links to the source code of documented Python objects (y/n) [n]: y
githubpages: create .nojekyll file to publish the document on GitHub pages (y/n) [n]: y
Create Makefile? (y/n) [y]: (leave as is, press enter/return)
Create Windows command file? (y/n) [y]: (leave as is, press enter/return)

(b) Generate reStructuredText (3)

1. Contents (1)
   The contents page should show q2 and the basic module inside it.
2. Basic Module (2)
   Add documentation to the binary search function.
   It should include a brief description, input parameters, and return details.
   The function declaration should indicate the data types for input and return.

Q3. Heaps in Python (Sphinx & Python) (15 + 10B)

You need to document the heap class and member functions (file heap.py has been provided in inlab06res/q3). Configuration stays the same as Q2(a), but the project name will be inlabq3.

The steps (and rubrics) are given below:

(a) Class description (5)

The description should include the utility/member functions of that data structure.

(b) Function descriptions (10)

Add documentation to all the member functions (insert, find, delete, print, reverse, etc.).
It should contain a brief description, input parameters, output parameters and return details (if any).

(c) Bonus (10)

Convert the Heap class to C++ and document it. Add to q1/DSA.cpp.

IMPORTANT: Make sure that the functions in the files are logically correct and run properly. If you find any errors, correct them in Q2 and Q3.
This is important for auto-grading.
For running the functions, you can call them by adding "if __name__ == 'main':" in the files.

Submission Instructions

As usual, create a folder named <roll-number>_inlab6, and compress it with the command (run it from the directory outside the folder):

tar - czvf <roll-number>_inlab6.tar.gz <roll-number>_inlab6/

The file structure should be as below (you can verify using the tree package):

```
<roll-number>_inlab6
    ├── q1
    │   ├── DSA.cpp
    │   ├── q1.pdf
    │   └── tester.cpp
    ├── q2
    │   ├── basic.py
    │   ├── make.bat
    │   ├── Makefile
    │   └── source
    │       ├── basic.rst
    │       ├── conf.py
    │       ├── index.rst
    │       └── modules.rst
    ├── q3
    │   ├── heap.py
    │   ├── make.bat
    │   ├── Makefile
    │   └── source
    │       ├── conf.py
    │       ├── heap.rst
    │       ├── index.rst
    │       └── modules.rst
    └── references.txt
```

Note: Mention all the links you use (apart from the ones in the resources) in references.txt.

May the Docs be with you!