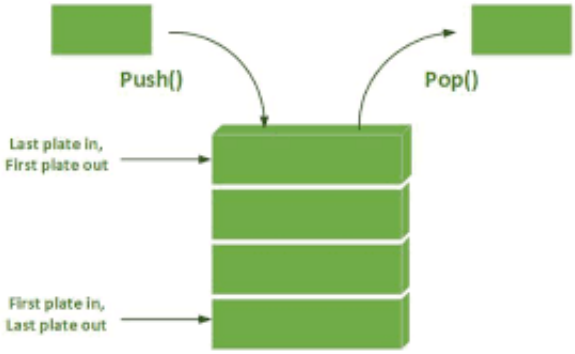
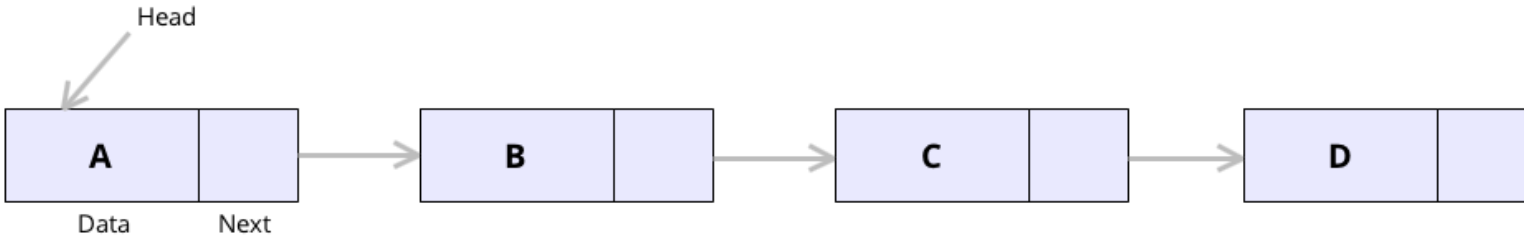
# Lab Assignment 6

**Due** Tuesday by 11:59pm **Points** 24 **Submitting** a file upload
**Available** Oct 20 at 8am - Oct 28 at 11:59pm 9 days

# LIFO Data Structure




# Application: Dynamic Stack Template




Make sure you have read and understood

- ***Unit modules 7 and 8***
- ***C++ Coding Style Guidelines*** (https://ohlone.instructure.com/courses/18987/files/3097729/download?download_frd=1)

before submitting this assignment. Hand in only one submission.

# Lab Assignment Objectives

1. Define a dynamic stack template.
2. Demonstrate how a stack class, as a last-in-first-out structure, can be implemented using a linked

2. Demonstrate how a stack class, as a last-in-first-out structure, can be implemented using a linked list class.

# Understand the Application

Define a class template that will create a dynamic stack of any data type.

# Testing Specification

Demonstrate the class template that you have written to create a dynamic stack. In summary, your should be able to handle pushing, popping and protecting against illegal operations attempted on an empty stack.

### *Example Test run*

```
/*
Pushing 14.1
Pushing 9.3
Pushing 25.7
Pushing 3.14
Pushing 2.75
Popping...
2.75
3.14
25.7
9.3
14.1

Attempting to POP again... The stack is empty.
*/
```

# What to Turn In

Hand in 3 files: **No zip files.**

- **DynStk.h :** dynamic stack template definition
- **a6cpp :** test driver file
- **Makefile**

# Tips and Requirements

1. Ensure that your solution is well organized. Provide a program header and comments to document and organize your source code. User defined methods need be documented.
2. Use a name guard on your header file.
3. Separate the stack template and driver source files: Multiple-file compilation is required.

4. Define and use named constants in lieu of literal values.
5. Provide a commented out copy of your program run. Enclose the run inside of multi-line comment delimiters so that your program will run in the grader test bed. Place the run after your program source code in the main.cpp file.
6. ***Initialize*** your submission files (i.e. **A**nn **O**hlone would submit files titled: **ao**DynStk.h and **ao**a6.cpp).

# Submission Resources

For more information on how to submit your assignment, please visit:

- **How do I submit an online assignment? Canvas Student Guide (https://community.canvaslms.com/docs/DOC-9539)**
- **Assignments Overview Canvas Video Guide (https://community.canvaslms.com/videos/1122-assignments-overview-students)**
- **Assignments Submissions Canvas Video Guide (https://community.canvaslms.com/videos/1121-assignment-submissions-students)**

# Submitting multiple files to an assignment

Your lab 6 assignment requires uploading more than one file; you should upload these 3 files as one submission. In this assignment you need to upload 1 **.cpp** file (a6.cpp using *yourinitials*a6.cpp) and 1 **.h** file (using *yourinitials*DynStk.h) and your solution Makefile. To add these files, the **Add Another File button** is clicked to **upload the two files one by one. Check to make sure that both files uploaded okay.** When finished click **Submit Assignment.**

# Questions?

Feel free to **ask** in the forum!

File Upload | Google Doc | Studio | Dropbox | Office 365

Upload a file, or choose a file you've already uploaded.

Upload File | Use Webcam

+ Add Another File

Click here to find a file you've already uploaded

Comments...

Cancel Submit Assignment

## Lab 6 Rubric

| Criteria | Ratings | | | Pts |
|---|---|---|---|---|
| On time submission | **4 pts** **On time** | **2 pts** **One day late** | **0 pts** **Two days late** | 4 pts |
| DynStk.h<br>Satisfies the DynStk.h requirements : naming guard, and method definitions to support a class template that will create a dynamic stack of any type. | **8 pts** **Full Marks** | | **0 pts** **No Marks** | 8 pts |
| Test driver file<br>a6.cpp : Satisfies the test run validation source statements as specified in the assignment. Demonstrates the class template with a driver program. | **6 pts** **Full Marks** | | **0 pts** **No Marks** | 6 pts |
| Commented out test run included<br>A copy of the program test validation run output is attached AFTER the source code in the a6.cpp test driver file (enclosed within comment delimiters). The run matches the source code submitted. The test run instantiates a dynamic stack object; demonstrates pushing and popping items using a dynamic LIFO data structure. | **4 pts** **Full Marks** | | **0 pts** **No Marks** | 4 pts |
| Coding style<br>Includes a program header that describes the application. Code is correctly formatted (i.e. follows code style rules). Class methods are documented. | **2 pts** **Full Marks** | | **0 pts** **No Marks** | 2 pts |
| | | | | Total Points: 24 |