Lab 8
Morse Code Generator

Topics

1. Classes
2. Inheritance
3. Dynamic Binding

Prelab Assignments

- If not done, finish reading assignment 7.
- Read about Morse Code. You will need a conversion chart for this lab.
- Review the concept of linked lists.

Interesting fact - The current Mars Science Laboratory has Morse code inscribed on its wheel! Scientists and Engineers on the ground use this pattern to track the path followed by the rover on Mars. For more information - http://www.nasa.gov/mission_pages/msl/news/msl20120829f.html.

Tasks:

Write a *message* class that should contain:

1. **Constructor** – an empty constructor and a parametric constructor.
   - Empty constructor should prompt user for the message.
   - Parametric constructor should take in the message when the object is declared and initialized.
2. **Destructor** – To free any memory that may have been allocated.
3. **Method: print** – This would print the message to the screen

Write a *morseCodeMessage* class that should extend the *message* class mentioned above:

1. **Translate** – function to translate your input message into Morse code (called during initialization).
2. Redefines the **print** method to print the message in both Morse code and text to the screen on two separate lines.

Write a *messageStack* class – A generic stack class.

1. **Constructor** – a parametric constructor. It can take one message object.
2. **Destructor** – To de-allocate memory.
3. **Methods**:
   - **Push** – You should follow LIFO (Last In First Out Technique)
   - **Pop** – You should follow LIFO
   - **Print stack** – to print all messages on the stack to the screen (beginning from the top of the stack). You should employ dynamic binding to call the appropriate print function depending on the message object type.

Push your code to your github repository.

Experiments:

Run various cases, using several messages. Make sure to test all your methods.

Deliverables:

1. Lab Report:
    1) The report should follow the lab report format posted on Blackboard.
    2) Results: show a few test cases. Include screenshots showing a few messages displayed.
    3) Discussion: Discuss your results. Describe any problems and interesting things that you may have encountered while coding the lab.
    4) Screenshot of your Github repository showing your commit.
    5) Append your source code with proper comments and indentation to the end of your report.

2. Demonstration:

    Show a few messages to the TA.

Grading:

Demo:	20	
Report:	40	
Code:	40	**(Well organized, properly commented and indented)**