UNIVERSITY OF CALIFORNIA, MERCED

BERKELEY · DAVIS · IRVINE · LOS ANGELES · MERCED · RIVERSIDE · SAN DIEGO · SAN FRANCISCO

SANTA BARBARA · SANTA CRUZ

CSE-165: Object-Oriented Programming
Lab 2

Spring 2024

Preliminary Notes

- Write separate file(s) for each exercise. Zip all your files together and submit the zip file to CatCourses.
- Your solution must be exclusively submitted via CatCourses. Email submissions will not be accepted. Pay attention to the posted deadline!

1 Stack (40 points)

Consider the file Stack.h. This file contains a definition of the Stack data structure from Chapter 4 of the textbook.

For this exercise, you must do two things:

1) Implement the methods of the Stack struct (including the initialize method of the Link nested struct) in Stack.cpp. Remember that the Stack is a first-in first-out data structure. You may use the textbook implementation as a reference. Do not modify Stack.h.

2) Write a program in creatingStack.cpp that declares an instance of Stack, fills it up with numbers (doubles), and then prints out the numbers. You must use the Stack member functions to do this.

The first line of input contains an integer N. This is followed by N lines, each containing a double. Read in these values and store them in your Stack object. Afterwards, print out the values. (Note, due to the nature of the stack, you will essentially be printing them in reverse order.) When the stack is finally empty, call the cleanup() function.

Example Inputs:
3
5.7
1.1
12.5
Output:
12.5
1.1
5.7

2 Pointer to Function (30 points)

One of the problems of the Stack is that if we make it handle generic types with void* pointers, they will not know how to delete the elements we insert in them. One way to solve this is to use a pointer to a delete callback function.

First, add to your Stack struct one more member variable that will hold a pointer to the following function prototype:

void deletecb(void* pt);

Next, a member function to set this pointer:

void setDeleteCallback(void (*delcb)(void* pt));

Finally, modify your cleanup method so that it traverses all elements of the stack and deletes the links using the delete callback for each stored void pointer. The user will be responsible for providing the delete callback and implementing it with the correct delete call after converting the void pointer argument to its correct type.

Your code will be tested using deleteCallback.cpp.

Example Inputs:
4
8.2
65.5
13.37
0.0
Output:
Deleting: 0
Deleting: 13.37
Deleting: 65.5
Deleting: 8.2

3 Constructor and Destructor (30 points)

Extend Stack again to incorporate constructors and a destructor.

Add two constructors: a default one that will create an empty stack, and another one that will receive an integer N and will build a stack of N elements, such that the first element is 1.0 and every subsequent element is incremented by 0.1.

Add a destructor that will delete the whole stack by using the delete callback function, as per the previous exercise.

Your code will be tested using constructorDestructor.cpp.

Expected output:

```
s1
Deleting: 2
Deleting: 1.5
Deleting: 1
Deleting: 0.5

s2
Deleting: 1.3
Deleting: 1.2
Deleting: 1.1
Deleting: 1

end
```