CSE 1384 - Stacks and Queues
Lab 8

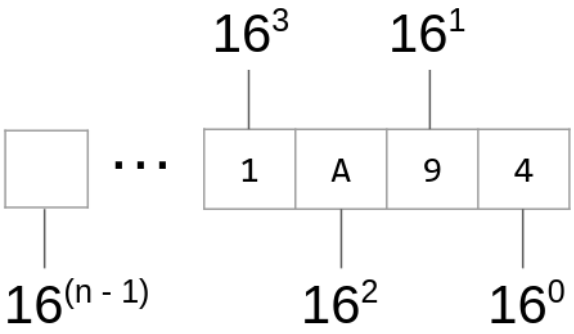
Objectives:

- Continue practicing past concepts
- Practice using stacks and queues in C++

Background:

This week's lab will involve hexadecimal to decimal conversion. As such, here's some background/explanation as to how that works if you haven't worked with it before.

Assume you have a four digit hexadecimal number – starting from the RIGHT-most digit, this will start at 16^0 and work your way up from 0, for every consecutive digit. So, four digits will have a LEFT-most digit that represents 16^3. This adjusts for every size, where your left-most digit ends up representing 2 brought to the power of the size - 1. A diagram of this concept is found below.



For hexadecimal, digits can take form in multiple values: 0-9 (coinciding with their usual decimal variants) and A-F (coinciding with 10-15).

Now, for conversion, you're adding the coinciding 16 exponential multiplied by the digit in the exponent's location. For a couple of examples:

Hexadecimal Number	Math Equivalent	Decimal Result
45E1	$16^3*4 + 16^2*5 + 16^1*14 + 16^0*1$	17889
AB	$16^1*10 + 16^0*11$	171
203	$16^2*2 + 16^1*0 + 16^0*3$	515
781F0	$16^4*7 + 16^3*8 + 16^2*1 + 16^1*15 + 16^0*0$	492016

Please find further resources on binary to decimal conversion here:

- Hexadecimal to Decimal video
- Converter (to double check your numbers)

Assignment:

You'll be given a starting point of `stack.h` and `node.h` – your job is to complete all of the functions in a `stack.cpp` file of your creation. Do NOT change either header file – your functions must adhere to the standards set in the headers. Note: the remove function returns an item. This should be returning the data that was contained at the node that was deleted. Create the add/remove functions in a way that adheres to STACK principles.

`main.cpp`:

This is where you'll need to use your stack to perform hexadecimal conversion. A lot of this design wise is left up to you (you can use functions to aid you, or not). However, you must fulfill these items:

- Ask for an initial hexadecimal number
- Display the finalized conversion result
- Ask the user if they'd like to enter another hexadecimal
    - Validate a yes/no answer
- Loop accordingly

Notice, the conversion itself will happen in main. You're using the stack to aid you in doing so. To be compatible with the Stack class, you should be taking your user input in as a string and iterating through each character to add an individual character to your stack.

A function in a provided `main.cpp` has been supplied to verify that your strings adhere to hexadecimal standards. You MUST use this function – user input should be verifiably a hex before conversion is attempted.

Hints:

In order to do exponents, you can use the cmath library through: `#include <cmath>.`

Details on the function you'll need to use can be found here.

Code / math related hints on the following page!

I recommend something like this to isolate what number value you'll need to multiply by the 16 exponential value. But, you can find your own solution. Just keep in mind that ASCII has a number associated with each character that C++ uses as a basis for the `char` datatype.

```
// checks the current letter's value
// 0-9
if(letter >= 48 && letter <= 57)
{
	// corrects to a 0-9 value from the ASCII char (48 --> ASCII 0)
	int num = int(letter) - 48;

	// you can use the number from here...
}

// A-F
else
{
	// corrects to a 10-15 value from the ASCII char (65 --> ASCII A)
	int num = int(letter) - 55;

	// you can use the number from here...
}
```

Milestones:

You must receive a check-off for each item to receive full credit on the lab. You may show the milestones incrementally, or receive a check off for multiple in one demo depending on your progress.

Examples of each milestone are included.

Milestone 1:

Stack class *completion*. SHOULD function with my provided `node.h` with NO changes to it or `stack.h`. **MUST BE SHOWN SEPARATELY.** Must show:

- Works with my test file! `testStack.cpp` has been provided to test your class – your stack.cpp should work with no changes and present the same behavior
- (this is all you'll use the file for … proceed with `main.cpp` after this point)

```
Enter a word: Maggie May Neal

laeN yaM eiggaM
```

Milestone 2:

`main.cpp` setup and looping. Must show:

- Asking for user input
- Working properly with my validation function to verify they've entered in a hex number
    - MUST show the looping works to show you didn't alter the function
- Asking if the user would like to continue
    - And looping behavior if they said yes
    - Exiting if they said no

```
Welcome to hexadecimal to decimal conversion.

Enter in a hexadecimal number to convert: AB1234

Would you like to enter another hexadecimal number (yes/no)? yes

Enter in a hexadecimal number to convert: 45a

Error. An invalid character was present in the hex number.
Please try again with characters 0-9 and A-F only.

Enter in a hexadecimal number to convert: ghr

Error. An invalid character was present in the hex number.
Please try again with characters 0-9 and A-F only.

Enter in a hexadecimal number to convert: 34F9C

Would you like to enter another hexadecimal number (yes/no)? no

Good-bye!
```

Milestone 3:

Error checking. Must show:

- Validate yes/no input in a LOOP when asking if the user would like to continue

```
Would you like to enter another hexadecimal number (yes/no)? YES

That's not a valid response. Try again.
Would you like to enter another hexadecimal number (yes/no)? fg

That's not a valid response. Try again.
Would you like to enter another hexadecimal number (yes/no)? no

Good-bye!
```

Milestone 4:

Put it all together and convert. Must show:

- Taking in a string
- Displaying a CORRECT conversion – to help illustrate, student's must show the following numbers:
    - AB1234 → result is 11,211,316
    - 65 → result is 101
    - 90F1 → result is 37,105
    - 4 → result is 4
    - FFFF → result is 65,535
- FULL process working together

```
Welcome to hexadecimal to decimal conversion.

Enter in a hexadecimal number to convert: 78E1
Your number converted is: 30945

Would you like to enter another hexadecimal number (yes/no)? yes

Enter in a hexadecimal number to convert: 1876
Your number converted is: 6262

Would you like to enter another hexadecimal number (yes/no)? no

Good-bye!
```

Comment Block:

Your code should contain a comment block at the top containing information on who wrote the code, what the assignment is, when it is due, etc. Here is an example of a good comment block to put:

```
/*
 Name: <your name>                          NetID: <your netID>
 Date: <current date>                       Due Date: <enter in due date>

 Description: <What is the program?>
*/
```

Deliverables:

- C++ code
    - Stack class (.h and .cpp files)
    - Main file (.cpp file)

Point Breakdown:

(100 points total)

A submission that hasn't received a check-off will not be considered for grading.

- 25pts - milestone 1
    - This milestone MUST be shown – you cannot just finish the lab and only show the last milestone. At bare minimum, milestone 1 and the last milestone must both be shown separately to ensure the class fits the Stack header provided
- 15pts - milestone 2
- 10pts - milestone 3
- 40pts - milestone 4
- 10pts - programming style *
- (PENALTY) 15pts - changing the code given to you (`node.h, stack.h`, OR any of the existing code in `main.cpp`)

* Programming style includes good commenting, variable nomenclature, good whitespace, etc.