CSCI 2270 - Data Structures and Algorithms

Instructor: Hoenigman / Jacobson
Recitation 3

In this recitation, you will be learning about dynamic memory allocation and an algorithm called array doubling. Arrays are great; they're simple and easy to use. They also have a big limitation in that they have a fixed size. If you need to store more data than you have room for in your array, you often need to create a new array with more space and copy over your values from your original array. Some languages provide interfaces such as Vectors and Lists that offer dynamic re-sizing. But, in other languages, arrays are still your best option.

First, a little background on memory allocation. We will also be discussing this in lecture also.

Static Memory Allocation:

Static memory allocation refers to the process of reserving memory at compile time, before the associated program is executed. Nothing can be added to it; it's fixed.

For example:

```
int array[10];    //declaring an array of 10 integers on the stack.
```

Dynamic Memory Allocation:

Dynamic memory allocation refers to the process of reserving memory at run time. The keyword new attempts to allocate enough memory in free storage for the new data.

```
int a = 10;
int* array = new int[a];              //declaring an array of size a on the heap
```

You use dynamic allocation when you don't know the size of an array at compile time. For example, imagine you're generating an array that contains all unique words in a document and you don't know ahead of time how many unique words there are. You may need to allocate memory for larger and larger arrays as more and more words are read in. This allows for a change in size.

Releasing dynamically allocated memory:

Any memory dynamically allocated with new must be released with delete to avoid a memory leak. The delete operator can be used in *two ways*: one for de-allocating memory allocated to arrays with new and the other for deleting single objects.

```
int* p_var = new int;
delete p_var;          // deleting the memory that the pointer is pointing to; is now free or available
p_var = nullptr;       // setting the pointer to null for safety.

int* p_array = new int[50];
delete [ ]p_array;
p_array = nullptr;
```

What is a memory leak in C++?

A memory leak occurs when a piece of memory that was previously allocated by a programmer is not properly de-allocated by the programmer. Even though that memory is no longer in use by the program, it is still "reserved", and that piece of memory can not be used by the program until it is properly de-allocated by the programmer. That's why it's called a memory leak – because it's like a leaky faucet in which water is being wasted, only in this case it's computer memory.

What problems can be caused by memory leaks?

The problem caused by a memory leak is that it leaves chunks of memory unavailable for use by the programmer. If a program has a lot of memory that hasn't been de-allocated, then that could really slow down the performance of the program. If there's no memory left in the program because of memory leaks, then that could of course cause the program to crash.

Array doubling algorithm

One of the most common ways to increase the size of an array is to generate a new array that is *twice the length* of the current array. Then, copy all values from the current array into the first half of the new array, and free the memory associated with the current array. In C++, we use dynamic memory allocation for this process.

Pseudocode

```
int* a = new int[length]                  // allocated on heap
int* array_doubled = new int[length * 2]      // create new bigger array.
for (i = 0 to a.end)
   array_doubled [i] = a[i]                              // copy values to new array.
delete [] a                               // free old array memory, w/o this step = mem leak
a = array_doubled                          // now a points to new array.
```

In class work:

You are given an array, its' size or length, and the number of times to double the array. Your task will be to find the expected output of the array after the doubling takes place? For the given array: A = {0,2}, what is the output when doubling = 1. See the above psuedocode for assistance. Show your TA the state of the new array at every iteration.

Recitation 3 Programming Assignment

There is a link on Moodle to a Recitation 3 Programming exercise. In the quiz, you are asked to implement an array doubling algorithm an arbitrary number of times using dynamic memory allocation.

You have until **Sunday at 5pm** to complete the exercise.