**CSCI 1411 – Lab 09 – Pointers and Dynamic Arrays**

**Goals:**

* Searching through arrays
* Sorting arrays with bubble and selection sorts

**Development Environment:** (all students must use Visual Studios)

**Skills:** pointers, dynamic arrays, searching sorting, functions

**Reading**: Chap 9

**Deliverables:** 1) This lab with 3 screen shots 2) lastnameFirstLab09.cpp

**Part I – Skills Practice (5 points)**

* Open a new project in Visual Studios. Call the project Lab09. Don’t forget to uncheck the Pre-compiled headers and Security Development Lifecycle checks. Then check the box for an empty project

You will be creating 3 files called lab09a.cpp, functions.h and functions.cpp

* In functions.h.

**#ifndef FUNCTIONS\_H**

**#define FUNCTIONS\_H**

**int\* allocateArray(int arraySize);**

**int\* increaseArray(int\* array, int &currentSize, int IncreaseBy);**

**void listArray(int array[], int size);**

**#endif**

* In functions.cpp

**#include <iostream>**

**#include "functions.h"**

**using namespace std;**

**int\* allocateArray(int arraySize)**

**{**

**int \*arrayPtr;**

**arrayPtr = new int[arraySize];**

**return arrayPtr;**

**}**

**int\* increaseArray(int\* arrayPtr, int &currentSize, int increaseBy)**

**{**

**int \*newPtr;**

**//create new array of larger size**

**newPtr = new int[currentSize + increaseBy];**

**//copy contents from old to new**

**for (int i = 0; i < currentSize; i++)**

**{**

**newPtr[i] = arrayPtr[i];**

**}**

**//intialize additional cells to -1**

**for (int i = currentSize; i < (currentSize + increaseBy); i++)**

**{**

**newPtr[i] = -1;**

**}**

**currentSize = currentSize + increaseBy;**

**delete[] arrayPtr;**

**return newPtr;**

**}**

**void listArray(int array[], int size)**

**{**

**for (int i = 0; i < size; i++)**

**cout << array[i] << " ";**

**cout << endl;**

**}**

* In Lab09a.cpp

**#include <iostream>**

**#include "functions.h"**

**using namespace std;**

**int main()**

**{**

**int \*arrayPtr;**

**int size = 3, inc = 2;**

**arrayPtr = allocateArray(size);**

**for (int i = 0; i < size; i++)**

**arrayPtr[i] = i;**

**listArray(arrayPtr, size);**

**arrayPtr = increaseArray(arrayPtr,size, inc);**

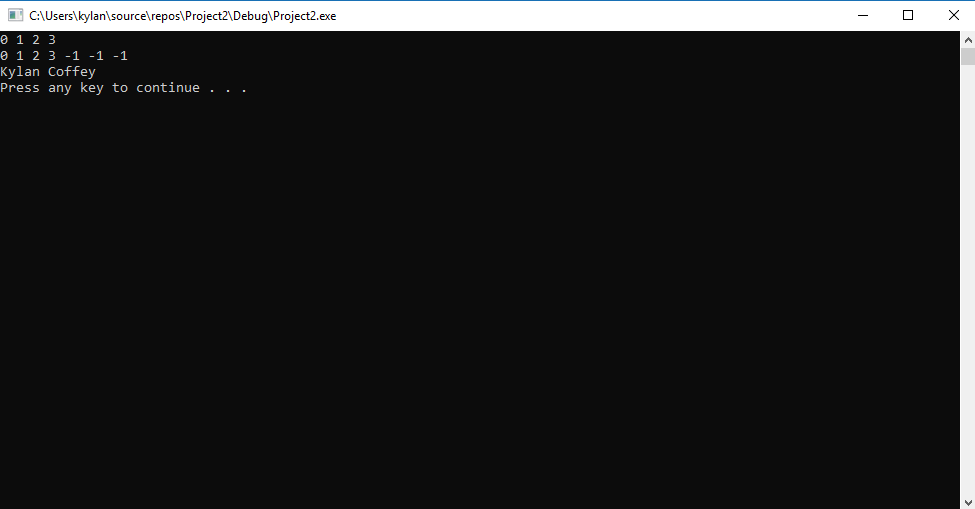
**listArray(arrayPtr, size);**

**return 0;**

**}**

* **Run** it with size = 4 and inc = 3. Take a **screen shot** of the successful output and place it below. Replace this output with your output (and 2 different cars) For a Windows 10 screen shot: Alt key + PrtSc key. Then Ctrl + V to paste. For Mac: Shift + Command + 4. You will not credit unless you have a successful screen shot with your name in the output.

**Part II – Use of Visual Studios Debugger (5pts)**

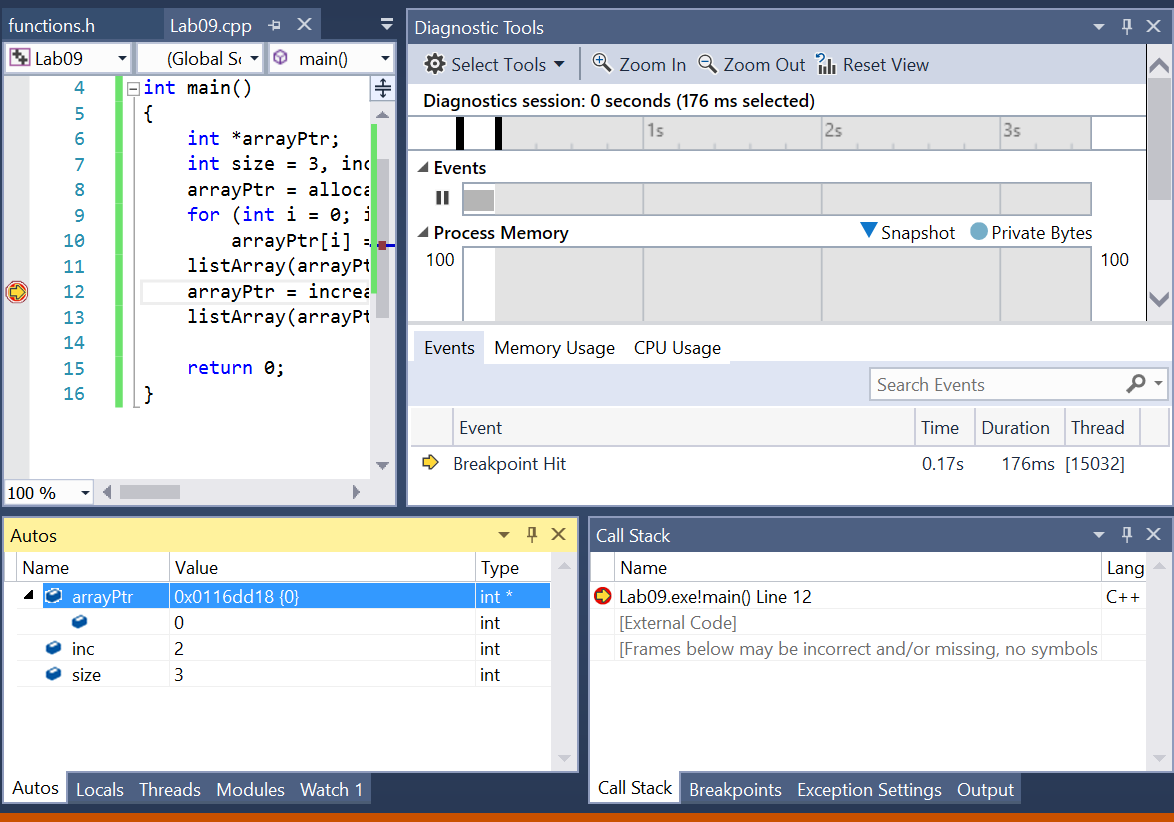


For this section, we are going to use the Visual Studios Debugger. You should use the debugger whenever you are using pointers in order to ensure that you are not going out of bounds and are pointing to the correct variables.

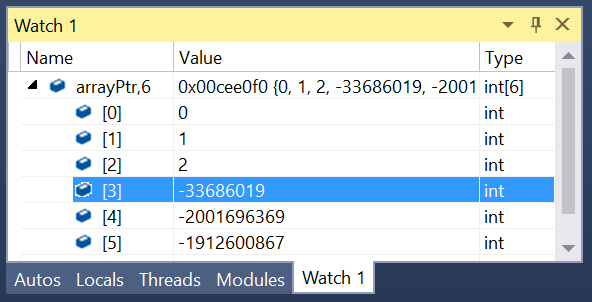
* Go to the Lab09a.cpp file
* Click on the number to the left of the line

**arrayPtr = increaseArray(arrayPtr,size, inc);**

* This will highlight the line
* Right Click. Go to Breakpoint-Insert Breakpoint
* Build->Re-Build Solution
* Debug->Start Debugging
* It will run the first part, and you will see a shell window open that starts printing the initial numbers. Now it is stopping at the line that you set a breakpoint.
* This should open window frames that look something like this



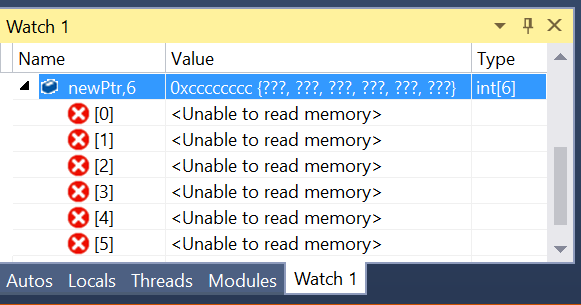
* Let’s focus on the Autos window. Note that inc and size have the value that you set at the beginning. ArrayPtr has a value starting with 0x which says that it is a hexademical number. This is a very long number (you’ll learn more in CSCI 1510) that denotes a starting memory address that the pointer “points to”. Note that the arrayPtr has an arrow before it, meaning you can expand it to see more information. In this case, the first value it points to is zero.
* Now what if we want to see the other values of arrayPtr?
* In the Autos tab, right click on arrayPtr and go to Add Watch. This places it in the Watch Window. Double click on the word arrayPtr, and change it to watch 6 positions. By changing arrayPtr to arrayPtr,6. It should look similar to this



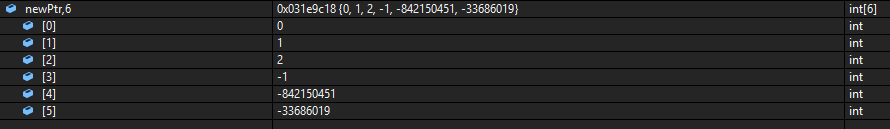
* Notice that you had set the values of the first 3 places inside Lab09a.cpp (at the for loop). You did not set the value for arrayPtr[3],[4], or [5], so that’s why it is whatever was in that memory position.
* Now we have a choice of going to the next statement, or stepping over the next statement. Hover your cursor over the symbols in the top right to see this (or they are also listed under Debug)



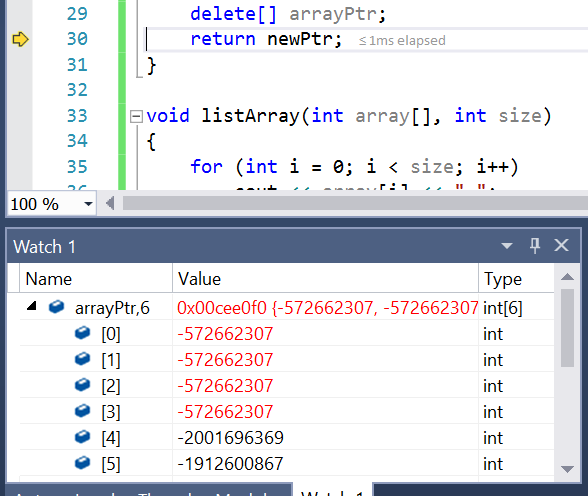
* First Step Into. That will take you into the increaseArray function
* Step over the **int \*newPtr;**
* Then you will see down in Autos that there is a newPtr entry. Right Click and Add Watch
* In the Watch window, double click on newPtr and change to newPtr, 6. Hit the enterkey

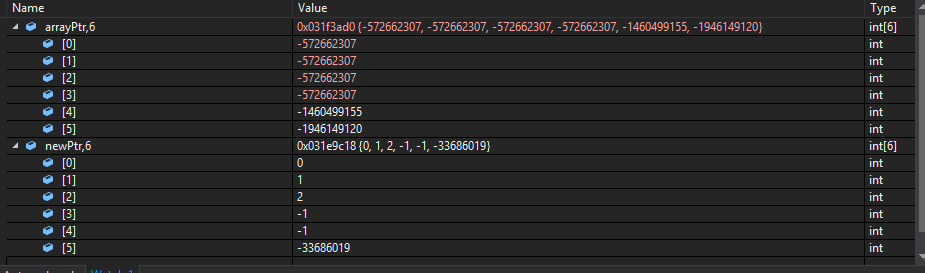


* **It is unable to read the memory, because if you notice at this point in the code we have not yet done the new, giving values.**
* **Step Over the newPtr = new int[currentSize + increaseBy];**
* Keep stepping over the for loop, noting that the value of newPtr[0], [1], etc. will change as you go through the loop.
* Take a **screen shot** after the values of newPtr [0], [1], [2] are filled. Place that below



* When you Step Over the **delete[] arrayPtr;** note that addresses go red, because this variable is no longer assigned in the program (because you removed the entire array from memory with delete[])





* Go to the stop icon and stop debugging. You can then click on the entire line again and Breakpoint->Delete Breakpoint.

It takes some practice to know how and when to use the debugger, but the above exercise should help. Please try this on your own in future programs, especially those dealing with pointers and memory addresses.

**Part III - outline/pseudo-code/algorithms (5 pts)**

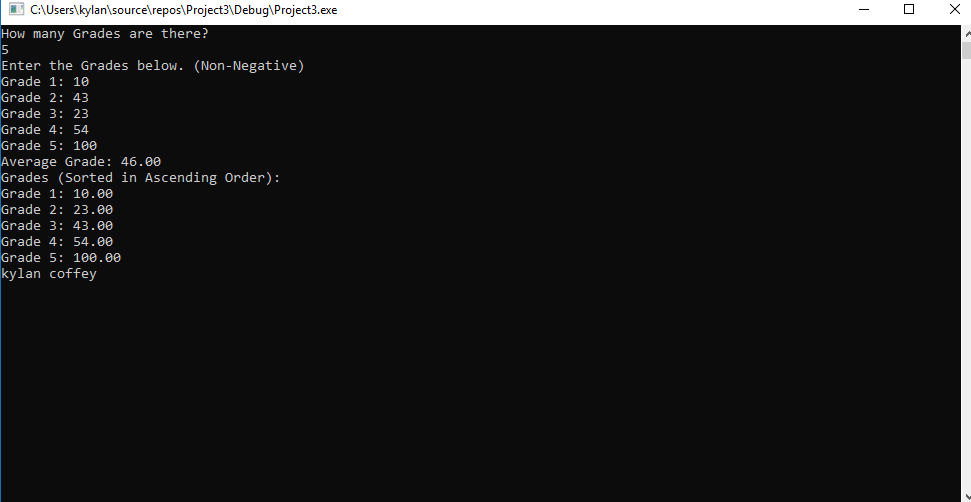
* Work with your lab partner to write an outline in comments and psuedocode to complete the program in part IV. Use plain English for your outline.
* You should have one file broken into sections with comments. Each .h and .cpp should be in a separate section. Every program you write should have the following block at the top in comments. Make sure to fill in the Name, Class, Description and Lab Partner at the top of the file. Ensure your status is accurate.

/\* Name:  
Class: CSCI 1411-00X  
Description: [fill in description]

Lab Partner:  
Status: successfully compiled and run on csegrid [if it doesn’t run or meet all of the requirements, list the actual status!/\*

**Part IV -Average Scores. (10 pts)**

* Write a program that dynamically allocates an array large enough to hold a user defined number of test scores. Once all the scores are entered, the array should be passed to a function that sorts them in ascending order. Another function should be called that calculates the average score. The program should display the sored list of scores and averages with appropriate headings. Use pointer notation rather than array notation whenever possible. Input validation: Do not accept negative numbers for test scores.
* Run the program and take a **screenshot** of the output. Place it below



* Turn in yourlastnamefirstLab09.cpp to canvas along with this lab with 3 screenshots.