COSC 1410
Spring 2016
Assignment 11: Pointers and Dynamic Storage

[1] **Objective**: This assignment will serve as an exercise for using pointers and dynamic storage. You will be using the **new** operator to create storage space (without a static variable name) to store data. You will also be using **delete** to remove data stored in the dynamic storage.

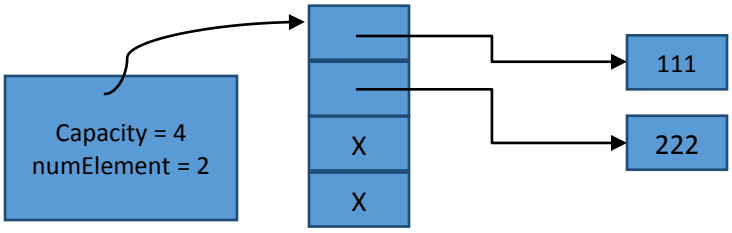
[2] **Description**: In this assignment you will implement a truly dynamic array to store integer values of a list. In C++, you can use **new** to create a dynamic array of a size that is determined at run time. This is better than the static array that you must specify the size at compile time. However, the size of the dynamic array is fixed once it is created (at run time). We are going to implement an even more dynamic array such that we can add new elements into the array (or list) without limit theoretically. How is this possible? You create a dynamic array of an initial size first. Whenever it is not enough, replace it with an array twice the size. That will last for a while.

The following pseudocode shows how to replace an array of size N with a new array of size 2N. This is possible because we are using dynamic storage, not a static array.

```
int **list; // no class is used in this example
list = new int*[N]; // allocating an array of size N
Store values in list
int **newlist;  //
newlist = new int*[2*N]; // allocating a new array of size 2N
copy the values from old list to new list
delete []list;
list = newlist;
```

Keep in mind that you have to copy the values in the smaller array to the larger array before deleting the smaller one. Similar technique can be used when replacing a larger array with a smaller one.

You have to define a class called dynArray. If you know how to use constructor, you may use a constructor. Otherwise use a function to initialize the array to an empty array. The dynArray must have the following three members: (1) An array of pointer to integer (2) The physical size of the array (initially 2) aka array_capacity, and (3) The actual size of the list (initially 0) aka number_of_elements.



You need the following six additional methods:

`showArray();` // display all the array elements. Also print out array capacity and actual size

`addElem(int);` // add an integer at the end of the list. The value of the integer will be `actual size of the list+1.`

`removeElem();` // remove the last integer from the end of the list. Show an error message if there is no more elements to remove from the array.

`doubleSize();` // private function to double the size of the array when actual size == array capacity. Show a message whenever you are doubling array size.

`halfSize();` // private function to reduce the size of the array to half when actual size == array_capacity/4. Show a message whenever you are reducing array size.

`int menu();` //private function to show the menu for user input. The menu() should have the following options: 1. Add element 2. Remove element 3. Show array 4. Quit. Once the user enters his/her choice, the corresponding function should be called.

For this assignment, we shall use an initial array size INIT_SIZE = 2.

Requirements:

1. You are not allowed to use pre-allocated (static) arrays.
2. You must use dynamic arrays. You are not allowed to create a dynamic array with a large size, because it must only grow at the rate of the doubling of the Current Count. Also during removal operation, whenever the actual size becomes one-fourth of the array capacity, you should reduce the array size into half.
3. Make all of the naming of the functions, variables, and class members appropriate to their purpose and use.
4. You must define a class called *dynArray*.
5. You must delete all unused dynamic storage.
6. No global variables except the constants.
7. Call the `showArray`() function after adding or removing an element to understand the impact.

[3] **Input**: Write a main function <u>like</u> this to test the class.

```
int main (){
dynArray a;
      while(1){
          int choice = a.menu();
          if(choice==4) break;
      }
      system("pause"); // for PC users
      return 0;
}
```

[4] Output: The output should show how and when the dynamic arrays change in size. Sample output is shown in the next page. I used three columns to save space.

[5] **Due:** May 2, 2016. No late assignment accepted after this day.

```
1. Add element
2. Remove element
3. Show Array
4. Quit
Enter choice:1
Array[0] = 1

Physical Array Size = 2
Number of Elements  = 1

1. Add element
2. Remove element
3. Show Array
4. Quit
Enter choice:1
Array[0] = 1
Array[1] = 2

Physical Array Size = 2
Number of Elements  = 2

1. Add element
2. Remove element
3. Show Array
4. Quit
Enter choice:1
***   Array size doubled to 4
Array[0] = 1
Array[1] = 2
Array[2] = 3

Physical Array Size = 4
Number of Elements  = 3

1. Add element
2. Remove element
3. Show Array
4. Quit
Enter choice:1
Array[0] = 1
Array[1] = 2
Array[2] = 3
Array[3] = 4

Physical Array Size = 4
Number of Elements  = 4

1. Add element
2. Remove element
3. Show Array
4. Quit
Enter choice:1
***   Array size doubled to 8
Array[0] = 1
Array[1] = 2
Array[2] = 3
Array[3] = 4
Array[4] = 5

Physical Array Size = 8
Number of Elements  = 5

1. Add element
2. Remove element
3. Show Array
4. Quit
Enter choice:5
Wrong input. Enter again.
1. Add element
2. Remove element
3. Show Array
4. Quit
Enter choice:2
Array[0] = 1
Array[1] = 2
Array[2] = 3
Array[3] = 4

Physical Array Size = 8
Number of Elements  = 4

1. Add element
2. Remove element
3. Show Array
4. Quit
Enter choice:3
Array[0] = 1
Array[1] = 2
Array[2] = 3
Array[3] = 4

Physical Array Size = 8
Number of Elements  = 4

1. Add element
2. Remove element
3. Show Array
4. Quit
Enter choice:2
Array[0] = 1
Array[1] = 2
Array[2] = 3

Physical Array Size = 8
Number of Elements  = 3

1. Add element
2. Remove element
3. Show Array
4. Quit
Enter choice:2
***   Array size reduced to 4
Array[0] = 1
Array[1] = 2

Physical Array Size = 4
Number of Elements  = 2

1. Add element
2. Remove element
3. Show Array
4. Quit
Enter choice:2
***   Array size reduced to 2
Array[0] = 1

Physical Array Size = 2
Number of Elements  = 1

1. Add element
2. Remove element
3. Show Array
4. Quit
Enter choice:2
***   Array size reduced to 1

Physical Array Size = 1
Number of Elements  = 0

1. Add element
2. Remove element
3. Show Array
4. Quit
Enter choice:2
You have no element in the array.
Nothing to remove
1. Add element
2. Remove element
3. Show Array
4. Quit
Enter choice:4
Quiting... Press any key to continue . . .
```