**Exercise 1.** Download and run the programs: dynamicArray.cpp and memLks.cpp by trying different inputs, and save your outputs in a file. Try to understand how the routines work and what is meaning of the outputs is. Write your comment below:

dynamicArray.cpp allows the user to create arrays using functions even though the local variables of functions are deleted after its execution. However, in this program the values of the array are not deleted because the use of dynamic memory allocation in arrays allows the creation of arrays using functions.

memLks.cpp creates arrays of doubles using the capacity given by the user. The arrays are created using dynamic memory allocation. However, there is no delete operator for the dynamic arrays, therefore the program does not get back the memory allocated and the memory is lost.

**Exercise 2.** Determine the output of the following code segment. Please try to answer the questions with your knowledge before running these codes using the C++ compiler.

int main()  
{  
 char C[] = {'E','X','A','M','S','T','R','I','N','G','\0'};  
 cout << C << endl; // Output: EXAMSTRING   
 cout << strlen(C) << endl; // Output: 10  
 cout << C+3 << endl; // Output: MSTRING   
 char \* cp = C;  
 \*(cp+4) = 'B';  
 cout << C << endl; // Output: EXAMBTRING  
 cp = cp + 6;  
 cout << cp << endl; // Output: RING  
 C[4] = '\0';  
 cout << C << endl; // Output: EXAM  
}

**Exercise 3.** Quick Quiz (open book / open note).

1. The delete keyword is used to release memory during program execution.  
2. Given the declarations  
int a[ ] = {44, 22, 66, 11, 77, 33};  
int \* p = a;  
what is the value of p [2]? 66  
3. In Question 2, a is a dynamic array. (True or False) False  
4. Write a statement to deallocate the memory that was allocated to the array dubArray with the statement double \* dubArray = new double[100];

Delete [ ] dubArray;

**Exercise 4.** Finish and test the following three functions in the skeleton code below. Upload your answers in a file named InLab3.cpp.

1) function int\* append (int\*, int, int\*, int); which accepts two arrays and returns a new array by appending the second array to the first array.

2) function int\* merge (int\*, int, int\*, int); which accepts two sorted arrays and returns a new merged sorted array.

3) function void print (int\*, int, const char \*); which accepts one array and prints the data in this array on-screen in the original order.

// check your outputs for correctness.   
// MUST tested your code first before appending your code over here  
int main()  
{  
 int arrayA[] = {11,33,55,77,99}; // use other values for more tests  
 int arrayB[] = {22,44,66,88};  
 print(arrayA, 5, "Sorted array A: ");  
 print(arrayB, 4, "Sorted array B: ");  
 int\* arrayC = append(arrayA,5,arrayB,4); // arrayC points to the appended array  
 print(arrayC, 9, "Append B to A: ");  
 int\* arrayD = merge(arrayA,5,arrayB,4);  
 print(arrayD, 9, "Merge A to B: ");  
 delete [] arrayC;  
 delete [] arrayD;  
#ifdef \_WIN32 // \_WIN32 is used by Visual C++  
#if (\_MSC\_VER <= 1916) // check if it Visual Studio 2017 or earlier  
 system("pause");  
#endif  
#endif  
 return 0;  
}

int\* append(int\* arrayA, int sizeA, int\* arrayB, int sizeB)  
{

int\* arrayC = new int[sizeA + sizeB];  
  
for ( int i = 0; i < sizeA; i++ )  
 arrayC[i] = arrayA[i];  
  
int j = 0;  
for ( int i = sizeA; i < ( sizeA + sizeB ); i++ )  
{  
 arrayC[i] = arrayB[j];  
 j++;  
}  
  
return arrayC;

}

int\* merge(int\* arrayA, int sizeA, int\* arrayB, int sizeB)  
{

int\* arrayD = new int[sizeA + sizeB];  
  
int a, b;  
a = b = 0;  
  
for ( int i = 0; i < ( sizeA + sizeB ); i++ )  
{  
  
 if ( a == ( sizeA - 1) && b != ( sizeB - 1) )  
 {  
 arrayD[i] = arrayA[a];  
 }  
 else if ( arrayA[a] < arrayB[b] )  
 {  
 arrayD[i] = arrayA[a];  
 a++;  
 }  
 else  
 {  
 arrayD[i] = arrayB[b];  
 b++;  
 }  
}  
  
return arrayD;

}

void print(int\* array, int size, const char \* lable)  
{

cout << lable << " " ;  
for ( int i = 0; i < size; i ++ )  
{  
 cout << array[i] << " ";  
}  
  
cout << endl;

}  
// paste your outputs here to verify your routine is working.

\* EXECUTION SAMPLE:  
\* Sorted array A: 11 33 55 77 99  
\* Sorted array B: 22 44 66 88  
\* Append B to A: 11 33 55 77 99 22 44 66 88  
\* Merge A to B: 11 22 33 44 55 66 77 88 99  
\*  
\* Process finished with exit code 0