# CMPT 130: Lab Work Week 12

1. Given two arrays **A** and **B** with equal size **n**, write a C++ function that returns true if every element of A is found in B; otherwise returns false.

   In your function there will be an **if**-**statement** that compares an element in A with an element in B, how many times does, in the worst case, that **if**-**statement** get executed? Give your answer in terms of **n**. Give the complexity of your algorithm in Big-O notation. Classify your algorithm as linear, logarithmic, quadratic, or some other class of functions. Assume the arrays **A** and **B** are not sorted.
2. Repeat question 1 above but this time assume the array **A** is not sorted but the array **B** is sorted.
3. Repeat question 1 above but this time assume the array **A** is sorted but the array **B** is not sorted.
4. Repeat question 1 above but this time assume both the arrays **A** and **B** are sorted.
5. Write a C++ function named **c_str_sequentialSearch** that takes two arguments: a C-string **s** and a character **ch** and returns the index of **ch** in **s** if it is found or return -1 if it is not found. **Hint:-** All you need to do is to first determine the length of the C-string (say for example by calling the function **getc_str_Length** you implemented above) and then adapt the given **sequentialSearch** code for the data types.
6. Write a C++ function named **reverseSequentialSearch** that takes an array of integers, its length and searchValue as arguments and returns the last index of the searchValue in the array. If the search value is not found, your function must return -1.
7. Write a C++ function named **reverse_c_str_SequentialSearch** that takes a C-string and a searchCharacter as arguments and returns the last index of the searchCharacter in the C-string. If the searchCharacter is not found, your function must return -1. Hint: In the function; first compute length.
8. Write a C++ function named **c_str_BinarySearch** that takes four arguments: a sorted C-string **s**, a start index, last index, and a character **ch** and returns the index of **ch** in **s** if it is found or return -1 if it is not found. Use binary search algorithm.