CMPT125, Fall 2018

Homework Assignment 2
Due date: October 19, 2018

Submit homework, printed or written in readable handwriting,
to the assignment boxes in CSIL ASB9838.

** Total number of points is 105. Your grade will be min(your points, 100) **

** In questions 1,2,3 you need to write your code in C **
** Test your code by running it on different inputs **

1) [15 points] Write a function in C that gets an array A of length n of ints and returns a (new) array that contains all values in the A without repetitions.

```
// Gets an array A of length n
// Returns array containing all elements of A without repetitions
// We assume that the array has at least n elements
int* remove_repetitions(int* A, int n)
{
      // implement me
}
```

2) [30 points] In class we saw linear search and binary search. The implementations used loops. In this question you will implement the functions using recursion.

   (a) [15 points] Write in C a recursive version of the linear search algorithm.

```
// Returns the index of item in the array.
// If A does not contain item, returns -1
int linear_search_rec(int* A, int n, int item)
{
      // implement me
}
```

   (b) [15 points] Write in C a recursive version of binary search algorithm.

```
// Returns the index of item in the array.
// If A does not contain item, returns -1
int binary_search_rec(int* A, int n, int item)
{
      // implement me
}
```

3) [15 points] Recall the quick sort algorithm we saw in class.

```
void quick_sort(int* arr, int n)
{
      int pivot_ind; // the index of the pivot
      if (n > 1)
      {
            pivot_ind = rearrange(arr, n);
            // sort left part and right part recursively
            quick_sort(arr, pivot_ind);
            quick_sort(arr+pivot_ind+1, n-pivot_ind-1);
}
```

Implement in C the function *rearrange(int* arr, int n)* as discussed in class.
Use the element arr[n/2] as a pivot.

4) [30 points] We said in class (without proving it) that any sorting algorithm in comparison model requires at least n log(n) time In this question we will see examples where sorting can be performed in linear time

(a) [15 points] Write an algorithm (as pseudo code) that gets an array of length n that contains only numbers 1...100, and sorts it in linear time.
Explain your algorithms in words, and explain why the running time is O(n).

```
// Assume that the array has n elements allocated
// and all values are between 1 and 100
void sort100(int* A, int n)
{
      ...
}
```

(b) [15 points] Write an algorithm (as pseudo code) that gets an array of length n that contains all numbers from 0 to n-1, and sorts it in linear time.
Explain your algorithms in words, and explain why the running time is O(n).

```
// Assume that the array has n elements allocated
// and all values are between 0 and n-1
void sort_n(int* A, int n)
{
      ...
}
```

5) [15 points] Recall the Insertion Sort algorithm we saw in class.

```
InsertionSort(a, n): // a is an array of size n
for i=0…n-1
{
        /* Assertion: at the beginning of the iteration a[0..i-1] is sorted */
        // merge a[i] into a[0..i-1]
        j = i
        while (j>0  and  a[j-1]>a[j])
        {
                swap( &a[j-1] , &a[j] )
                j = j-1
        }
}
```

(a) [5 points] How many swaps will the algorithm make on input [2, 1, 4, 3, 7, 6, 5]?

(b) [10 points] Suppose that on some input of length n the algorithm makes S swaps. Prove that the total running time of the algorithm in O(n +S).