THE UNIVERSITY OF SYDNEY

INFO1910 Week 7 Tutorial

In this tutorial you will be writing a small c library for various functions related to int arrays. Note that you should write this code assuming that the integers may also be negative.

Question 1

Write max, min, and sum functions which take in an integer array and return the smallest, largest, and total of all the elements respectively.

```
int sum(int* array, int len)

int max(int* array, int len)

int min(int* array, int len)
```

1 Question 2

Write an average function, which returns the average value of all the integers in the array.

```
float average(int* array, int len)
```

2 Question 3

Write a contains_subsequence function. This function takes in a primary array, a secondary array representing a sequence of integers, and finds whether or not the given sequence appears in the primary array. If the array does contain the sequence, return the index at which it begins (if the sequence appears multiple times, return the index of the first time). Otherwise, return -1.

```
int contains_subsequence(int* array, int* sequence,
        int arr_len, int seq_len)
```

3 Question 4

Write a partial_sums function. The partial sum at a certain index of an array of integers is the sum of all elements of the array so far. The function takes in an array to read from, and an array into which the partial sums will be written. You may assume that the partial sums array will be at least as large as the original (make sure to set up an empty array of the right length when testing this function).

```
int partial_sums(int* array, int* partial_sums, int len)
```

4 Question 5

Write functions to flatten and un-flatten a two-dimensional array. Flattening a 2d array means to write the contents of the 2d array into a 1d array provided (the 1d array will have the same number of cells as the total number of cells in the 2d array). First write the elements of the 2d array's first row into the 1d array in order. Then write the elements of the second row, and so on. Un-flattenning the array is the opposite procedure; given a 1d array, reconstruct the 2d array which would have produced it when flattened. When testing, make sure to provide arrays of the right size and dimension count for data to be written into.

```
int flatten_2d(int** array_2d, int* array, int outer_len, int inner_len)
int unflatten_2d(int** array_2d, int* array, int outer_len, int inner_len)
```

5 Question 6

Write functions to flatten and un-flatten a three-dimensional array.

```
int flatten_3d(int*** array_3d, int* array, int dim1_len,
        int dim2_len, int dim3_len)
int unflatten_3d(int*** array_3d, int* array, int dim1_len,
        int dim2_len, int dim3_len)
```

6 Question 7

If you have been using a main function in the same source code file to test your functions, remove it. Create a header (.h) file declaring the existence of the functions in this file. Practise writing a separate main function in a separate file, including the header, and compiling the int array library along with your main program file so that the functions you have written can be used in other programs you may write in the future.