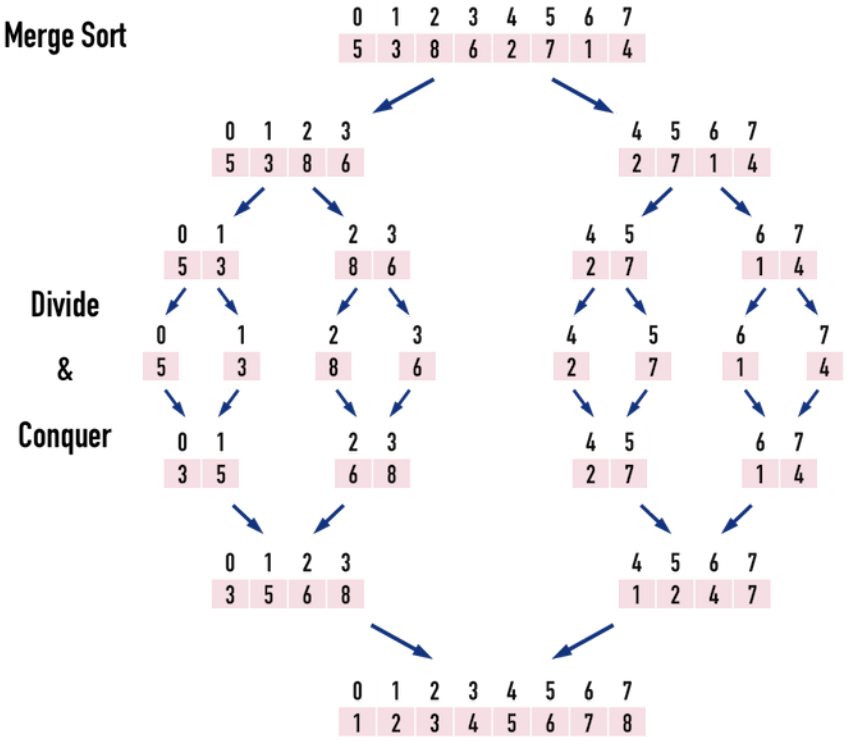
**hhCSCI 2520 Data Structures and Applications**
**Assignment Two - Sorting**
**Deadline: 23:55, APR. 13, 2020**
**Total Marks: 100**

**Submission:**
In this Assignment, you need to answer question 1 in one pdf file. For question 2, 3 and 4, you need to provide .c file for each question, which can be compiled and give correct answer, the name should be q2.c, q3.c and q4.c respectively.
Then compress all 4 files (one pdf file for q1 and three .c files for q2, q3 and q4 respectively) as one zip named as `your_student_id_assign2.zip` and submit it via blackboard.

**1. Merge Sort (20 marks)**

Suppose we have an integer array: {12, 34, 5, 86, 0, 87, 64, 158, 90, 56, 37, 84, 67, 73, 36, 91}, draw a figure to show the divide and conquer procedure.

This is an example for the figure:




**2. Quick Sort (20 marks)**

Write a C/C++ program to implement quick sort algorithm **recursively** on an array of integers, select the leftmost element as the pivot. Also count how many times **`quickSort`** is called, including the very first one.

Format of functions:

```
int quickSort( int * num, int start, int end );
```

where the return value is how many times **`quickSort`** is called.

**Sample Output:**

Enter how many numbers: 10
Input all integers: 20 19 25 29 13 40 47 32 41 50
Final result: 13 19 20 25 29 32 40 41 47 50
Called times: 11

Sample program is provided in **assign2-2.c**, you just need to finish the implementation of function `quickSort` in the tail of the file.

**3. Selection Sort (30 marks)**

Write a C/C++ program to implement selection sort algorithm on an array of strings to sort them alphabetically. Show results after each pass of sorting (See Sample Output).

**Note:** All the input strings have length of 4 letters. The input part is implemented in given code.

Format of functions:

```
void Selection_sort( char A[][5], int N );
```

where char A[][5] contains the N elements, and you need to use PrintA(A, N) to output the intermediate result.

**Sample Output:**

Enter how many strings: 5
Input all strings: abcd bcda bbcf fbac accm
Initial Array: abcd bcda bbcf fbac accm
After Round-0: abcd bcda bbcf fbac accm
After Round-1: abcd accm bbcf fbac bcda
After Round-2: abcd accm bbcf fbac bcda
After Round-3: abcd accm bbcf bcda fbac

Sample program is provided in **assign2-3.c**, you just need to finish the implementation of function `Selection_sort` in the tail of the file.

**4. Sort Three Distinct Keys (30 marks)**

Suppose you have an array of N elements, containing three distinct keys, "true", "false", and "maybe". Given an O(N) algorithm to rearrange the list so that all "false" elements precede "maybe" elements, which in turn precede "true" elements.

Format of functions:

```
void MySort( ElementType A[], int N );
```

where ElementType is defined as the following:

```
typedef enum { true, false, maybe } Keys;
typedef Keys ElementType;
```

and ElementType A[] contains the N elements.

**Sample Input:**

```
6
2 2 0 1 0 0
```

**Sample Output:**

```
false in A[0]-A[0]
maybe in A[1]-A[2]
true in A[3]-A[5]
```

Sample program is provided in **assign2-4.c**, you just need to append the implementation of function **`MySort`** to the tail of the program.