**ALGORITHMS & DATA STRUCTURES 2**
**Mid-Term Assignment**
**Back me up!**

## 1. BACKGROUND

Data is stored on hardware, and hardware can sometimes fail. One way to store data reliably is to store copies, which is called "data redundancy", so if something happens to the hardware storing one copy, a failure can be detected, and it could be recovered from elsewhere. A method of storing data in this way is RAID, which can stand for "Redundant Array of Inexpensive Disks" or "Redundant Array of Independent Disks". Here the idea is to distribute data storage among many pieces of hardware, such as hard disks; initially, this speeds up the reading and writing of data, but with no redundancy. To also introduce redundancy the data is copied and distributed on multiple disks.

There are different "levels" of RAID of which the most common are RAID 0 and RAID 1: the former involves distributing data to two disks without redundancy, and the latter copies in two disks introducing redundancy. We will use these two methods of data storage as inspiration; **you do not need to have any prior knowledge of RAID to attempt this assignment.**

In this assignment we will consider a set of scenarios based on a simplified version of RAID where we wish to store the contents of an array in two arrays. The first setting distributes the data of a single array into two arrays. After this we will consider two methods for storing copies of data from a single array into multiple arrays. In this assignment, in the first four tasks, we will mainly focus on methods to search the arrays for integer values. In the fifth task, which is more open-ended, you have the scope to invent a new scenario and describe how one can search data within it.

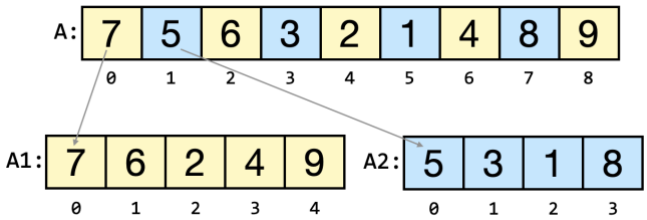
## 2. GUIDANCE FOR STUDENTS

The total marks available for this assignment is 100, broken into five tasks, each with smaller elements. In this assignment, a mark of 70 and above is considered excellent, 60 and above is very good, 50 and above is good, and 40 and above is satisfactory. A mark of 80 and above is reserved for those candidates that go beyond what is required of them in the course. There is scope to demonstrate this advanced knowledge in the final task of this assignment.

In your answers, being clear and concise is a good approach. If, in one of the questions, you are required to write only a pseudocode function, you do not need to include a detailed explanation of the pseudocode; at most short comments are more than sufficient. If, in other questions, you are asked to very briefly describe something, a few sentences are sufficient; several connected bullet points instead of prose are also sufficient. Finally, for brief explanations, a general guideline could be a sentence **at most** for each mark assigned to a question.

### 3. THE FIRST PART: THE R0 SEARCH ALGORITHM

The first scenario we consider is inspired by RAID 0. In this scenario we start with an array A of length N storing non-negative integers, and create two new arrays where the first array stores the values `A[i]` where `i` is an even number, and the second array stores the values `A[j]` where `j` is an odd number. The following diagram shows this scenario:




In this diagram we see that two new arrays, A1 and A2, are created and the values at even-numbered indices of A are copied to A1, and the values at odd-numbered indices are copied to A2.

The idea is now that instead of storing A, we store two separate arrays in a distributed manner. However, if we want to search the array A for particular values, we now need to search these two arrays and then relate the indices back to the original array. The next two tasks together give such a method to search the two arrays in this scenario.

**Task 1:** Consider the following pseudocode function:

```
function Find(key,B,M,j)
      for 0 <= i < M
            if(B[i] == key):
                  return j + 2*i
      return -1
```

1. This function takes an array B, its length M, a non-negative integer `key`, and a non-negative integer `j` as inputs, and returns an integer. Briefly explain why if the input array B is one of the arrays (A1 or A2) created as above, this function will return the index where the value `key` is located in the original array A. In your explanation refer to the input `j`, and to what it corresponds in array A. [5 marks]
2. Write a pseudocode function called `RecursiveFind`, which is a recursive implementation of the function `Find`: it should take the same inputs as `Find` in addition to another integer `i`, and return the same integer as `Find`. HINT: the integer argument `i` should vary in the recursive function calls. [10 marks]
3. The worst-case running time of a recursive implementation of `Find` for an array B of length M is T(M) = T(M-2) + c, where c is a constant. Very briefly explain why the Master Theorem is not relevant here for computing an expression of T(M) in terms of M. [5 marks]

**Task 2:** Consider the following pseudocode function that describes the **R0 Search** algorithm:

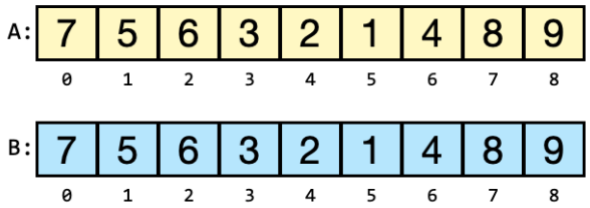
```
function R0(key,A1,A2,N)
      index = Find(key,A1,ceiling(N/2),0)
      if (index == -1):
            return Find(key,A2,floor(N/2),1)
      return index
```

In this pseudocode `floor(x)` and `ceiling(x)` are the mathematical functions that, respectively, give the largest integer smaller than or equal to x and give the smallest integer larger than or equal to x. For this algorithm address the following:

1. Identify, and describe very briefly in words, the **best-case inputs** and the **worst-case inputs**. Recall that there are four inputs to R0. [8 marks]
2. An expression for both the worst-case and best-case **running times** (or execution time) T(N), and describe the method by which you arrive at this expression. [8 marks]
3. The **growth function** of the worst-case and best-case running times T(N), i.e. a function that does not include constants or low-order terms, e.g. if f(N) = 5N+2, then the growth function is N. [5 marks]
4. The **Theta notation** for the worst-case and best-case running times T(N). In particular, find a set of constants c1, c2 and m0 for which T(N) is Θ(g(N)). [6 marks]

**3. THE SECOND PART: THE R1 SEARCH ALGORITHM**

The second setting is now inspired by RAID 1. In this scenario, given an array A of non-negative integers of length N, additionally a second array B also of length N is created; each jth element `B[j]` is then assigned the value `A[j]`, as we can see in the figure below.




The second array B is now introducing redundancy, which allows us to detect if there has been a hardware failure: in our setup, such a failure will mean the values in the arrays are altered unintentionally.

**From now on in this assignment we will assume that at most one array might have its values altered, but we do not know which one ahead of time.**

The goal of this part of the assignment is to write an algorithm to search for a non-negative integer in the array A. The second array B is there so that if a value is found in an element `A[j]`,

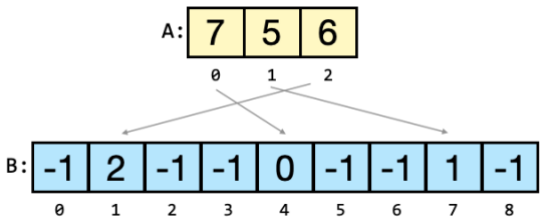
if this value does not match the value stored in `B[j]`, then there has been an error, and an appropriate error value should be returned.

**Task 3:** Complete the following:

1. Write a pseudocode function `R1(key, A, B, N)` that takes a non-negative integer key, and arrays A and B of length N as inputs: the function should return an index where the value `key` is stored in the same location in both A and B; it should return `-1` if the value cannot be found in either array; and it should return `-2` if there is an index `j` where `A[i]` is not equal to `B[i]`. [8 marks]
2. The **Theta notation** derive the worst-case running time T(N) of the algorithm `R1`. Describe the steps taken in your reasoning about what is the worst-case input and the resulting worst-case running time. [6 marks]

**4. THE THIRD PART: THE R1HASH SEARCH ALGORITHM**

In this next scenario, given an array A of non-negative integers of length N, a second array B of length $N^2$ is created: for each element `i` of A, the value `A[i]` is hashed to give an index `j` of B, and the index `i` is stored in `B[j]`. For the hashing function, given constants `a>0` and b, for each `i`, the value `A[i]` is hashed by the function `(a*A[i] + b) mod N`$^2$, which is equal to the index of B into which we store `i`. The following example demonstrates this:




In this example the hash function is `(3*A[i] + 1) mod 9`. For instance, the value 5 stored at index 1 in A is hashed to the value `(3*5 + 1) mod 9` = 7, and thus the index 1 is stored at index 7 in array B. In this example there were no collisions in the hashed values since all values in A are distinct. In general, for this setting, for all distinct values x and y assume that we use constants a and b such that the hashed value (a*x + b) mod $N^2$ is **not equal** to (a*y + b) mod $N^2$. If the array A stores the same integer multiple times, we resolve the collision in the hashed values through linear probing: to insert a value into B, if there is already a non-negative integer stored at a location `j`, we loop over the subsequent elements `j+1`, `j+2`, *etc* to find the next element storing the value `-1`.

**Task 4:** Complete the following:

1. Write a pseudocode function `R1Hash(key, a, b, A, B, N)` that takes non-negative integers `key`, `a` and `b` where `a` and `b` come from to the hashing function used to store indices in B; the arrays A and B of length N and $N^2$ respectively are also inputs: the function

should return an index `i` where the value `key` is stored in array A; it should return `-1` if the value cannot be found; it should return `-2` if there was an error in the data storage, i.e. one of the values was altered unintentionally in at most one of the arrays. Recall that we are assuming that the hashing function avoids collisions in the hashed values for distinct values. [8 marks]

2. Briefly explain your pseudocode, and how it returns the correct output in the three cases mentioned above. [6 marks]

**5. THE FINAL PART: DESIGN YOUR OWN SETTING**

**Task 5:** Devise your own setting for storing and searching the data in an array of non-negative integers redundantly. You may just describe the setting without having to give an explicit algorithm to explain the process by which data is stored. You should explain how hardware failures can be detected in your method. Once you have described the setting, complete the following:

1. Write a pseudocode function to describe an algorithm where the stored data can be searched for a value `key`: if the data is found, its location in the original array should be returned; `-1` should be returned if the data is not found; `-2` should be returned if there is a data storage error
2. Include a short commentary explaining why your pseudocode works
3. Describe the worst-case and best-case inputs to your search algorithm
4. Derive the worst-case and best-case running times for the search algorithm
5. Derive the Theta notation for the worst-case and best-case running times

Maximum word count for whole task: 750 words. The word count does not include the pseudocode for the search algorithm, any picture figures and any mathematical formula.

[25 marks]