ALGORITHMS & DATA STRUCTURES 2
Mid-Term Assignment
Houston, we have an error!

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 first three are RAID 0, RAID 1 and RAID 2: the first involves distributing data to two disks without redundancy; the second copies in two disks introducing redundancy; and the third also "adds" data together to compute a number called the parity, which checks if data has been altered. We will use these three 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 inspired by a simplified version of RAID where we store non-negative integers in arrays. The first setting distributes the data of a single array into two arrays. After this we will be inspired by RAID 2 and store sums of the integers in another array. After this we will use hashing to store copies of the original array. In this assignment, in the first four tasks, we will mainly focus on methods to search the arrays for integer values or errors. 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 and detect errors.

2. GUIDANCE FOR STUDENTS

SUBMIT A PDF OR YOUR WORK MAY NOT BE MARKED

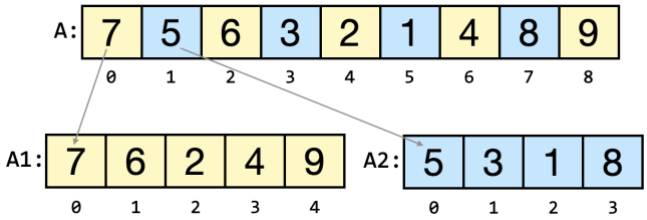
The total marks available for this assignment is 100, broken into five tasks, each with smaller elements. In this assignment, a mark of 70 and above is considered excellent, 60 and above is very good, 50 and above is good, and 40 and above is satisfactory. A mark of 80 and above is reserved for those 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:



There are now 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 that describes the **R0 Search** algorithm:

```
function R0(key,A1,A2,N)
     for 0 <= i < ceiling(N/2)
          if(A1[i] == key):
               return 2*i
     for 0 <= i < floor(N/2)
          if(A2[i] == key):
               return 2*i + 1
     return -1
```

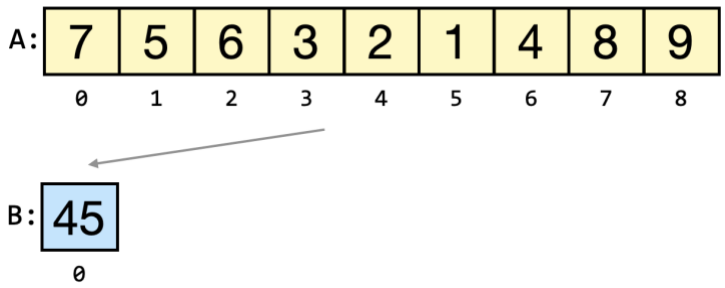
In this pseudocode `floor(x)` and `ceiling(x)` are the mathematical functions that, respectively, give the largest integer smaller than or equal to x and give the smallest integer larger than or equal to x. You may assume that calculating the `floor(x)` and `ceiling(x)` takes constant time. For this 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]

4. THE SECOND PART: THE R2 SEARCH ALGORITHM

The second setting is now inspired by RAID 2, which stores the "parity" of bits in another piece of hardware. To start this idea, we consider a simple case where, given an array A of non-negative integers of length N, additionally a second single-element array B is created. The array B will store the sum of all the integers in array A, as the following figure demonstrates:



The idea is that if there is an error in one of the two arrays that changes the values of the integers, the hope is that the value in B will not match the actual sum of all the integers in A.

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.

Task 2: Consider the following pseudocode functions that create such an array B for array A and the integer N, which is the length of the array A:

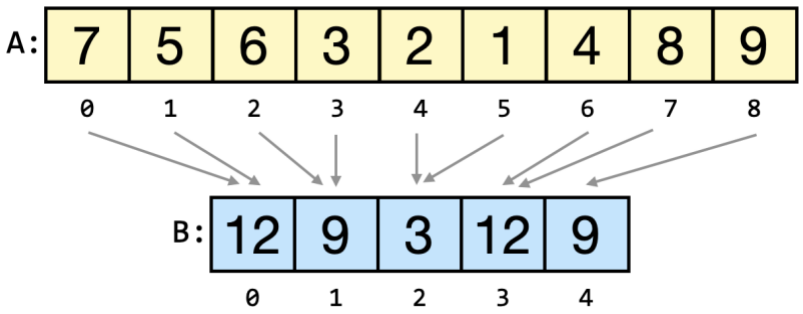
```
function Sum(A,left,right)
      if left > right:
            return 0
      else if left = right:
            return A[left]
      mid = floor(N/2)
      lsum = Sum(A,left,mid)
      rsum = Sum(A,mid+1,right)
      return lsum + rsum

function CreateB(A,N)
      B = new Array of length 1
      B[0] = Sum(A,0,N-1)
      return B
```

The function `CreateB` creates this second array using a function call to `Sum`. For the algorithm CreateB address the following:

1. Explain very briefly in words why the **best-case inputs** and the **worst-case** inputs are the same for CreateB. Recall that there are two inputs to `CreateB`. [6 marks]
2. Use the **Master Theorem** and to derive the **Theta notation** for both the worst-case and best-case **running times** (or execution time) T(N), and show your working and reasoning. [10 marks]

Building on the above, in a new scenario, given an array A of non-negative integers of length N, additionally a second array B is created; each element `B[j]` stores the value `A[2*j]+A[2*j+1]`. This works straightforwardly if N is even. If N is odd then the final element of B just stores `A[N-1]` 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. The hope is that if there is an error in A which changes the integer values then the sums in B are no longer correct and the algorithm says there has been an error; if there were an error in B the values would hopefully be incorrect

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 is now to write an algorithm to search for a non-negative integer in the array A, but also to check for errors in the arrays. If there has been an error determined from checking both A and B, and an appropriate error value should be returned. If no errors are detected then we determine if the integer is in A or not.

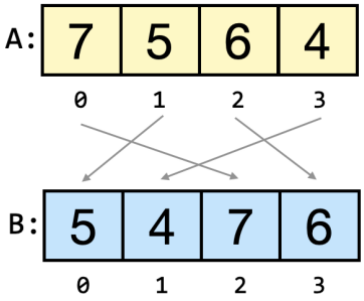
Task 3: Complete the following:

1. Write a pseudocode function `R2(key, A, B, N)` that takes a non-negative integer key, and arrays A and B, as well as the length of A, N: the function should return `-2` if the values in B do not correspond to the sums of values in A; if no error has been found the function should return the index if `key` is found in A, or `-1` if `key` has not been found. [8 marks]

2. The **Theta notation** derive the worst-case running time T(N) of the algorithm R2. Describe the steps taken in your reasoning about what is the worst-case input and the resulting worst-case running time. [6 marks]
3. Very briefly explain why not all errors that can occur for array A will be detected in the method above. [4 marks]

5. THE THIRD PART: THE R1 SEARCH ALGORITHM

In this next scenario, inspired by RAID 1, we will just duplicate all the given data using hashing. Given an array A of non-negative integers of length N, a second array B of length N is created: for each element `i` of A, the value `A[i]` is hashed to give an index `j` of B, and the value `A[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`, 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 4`. For instance, the value 5 stored at index 1 in A is hashed to the value `(3*5 + 1) mod 4 = 0`, and thus 5 is stored at `B[0]`. In this example there were no collisions in the hashed values since all values in A are distinct.

For this task we assume there are no repeated values in A, and a hash function is chosen so there are no collisions for distinct values.

To look for errors we hash all the values in array A to see if there is a match in the corresponding element of array B. If there is a mismatch between the two elements then an error is returned; if no errors are found, if a value is found then its index in A is returned.

Task 4: Complete the following:

1. Write a pseudocode function `R1(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 are also inputs: the function should return an index `i` where the value `key` is stored in array A if found; it should return `-1` if the value cannot be found; otherwise 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. [8 marks]

2. Briefly explain how the method above could be adapted to allow for repeated integers in array A. [6 marks]

6. 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]