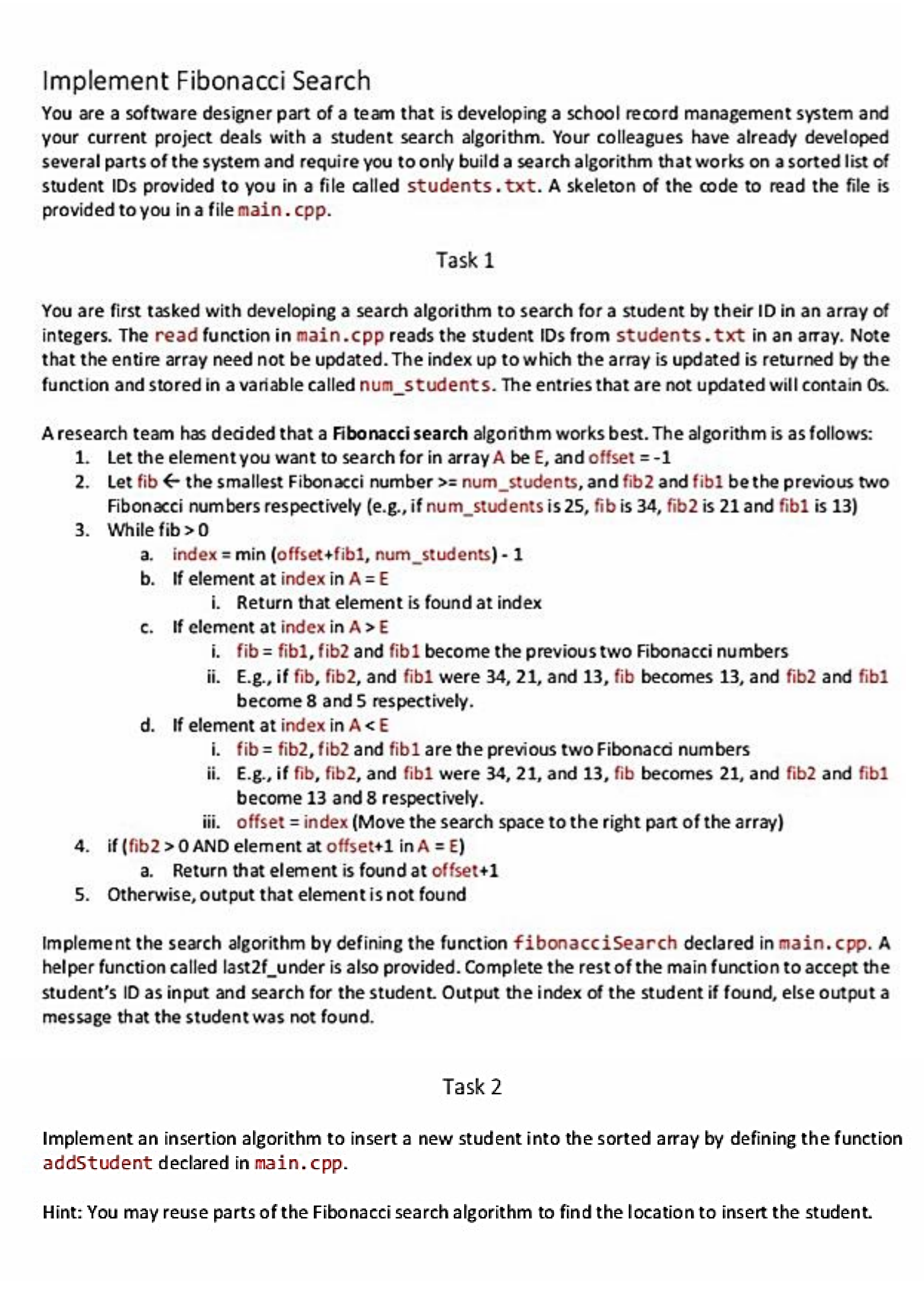


Implement Fibonacci Search

You are a software designer part of a team that is developing a school record management system and your current project deals with a student search algorithm. Your colleagues have already developed several parts of the system and require you to only build a search algorithm that works on a sorted list of student IDs provided to you in a file called `students.txt`. A skeleton of the code to read the file is provided to you in a file `main.cpp`.

Task 1

You are first tasked with developing a search algorithm to search for a student by their ID in an array of integers. The `read` function in `main.cpp` reads the student IDs from `students.txt` in an array. Note that the entire array need not be updated. The index up to which the array is updated is returned by the function and stored in a variable called `num_students`. The entries that are not updated will contain 0s.

A research team has decided that a **Fibonacci search** algorithm works best. The algorithm is as follows:

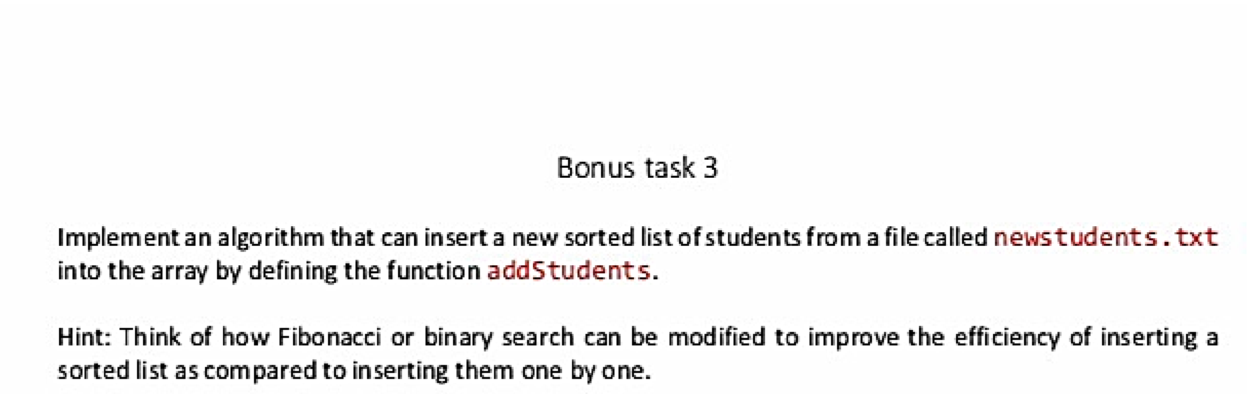
```
1. Let the element you want to search for in array A be E, and offset = -1
2. Let fib ← the smallest Fibonacci number >= num_students, and fib2 and fib1 be the previous two
   Fibonacci numbers respectively (e.g., if num_students is 25, fib is 34, fib2 is 21 and fib1 is 13)
3. While fib > 0
    a. index = min (offset+fib1, num_students) - 1
    b. If element at index in A = E
        i. Return that element is found at index
    c. If element at index in A > E
        i. fib = fib1, fib2 and fib1 become the previous two Fibonacci numbers
        ii. E.g., if fib, fib2, and fib1 were 34, 21, and 13, fib becomes 13, and fib2 and fib1
            become 8 and 5 respectively.
    d. If element at index in A < E
        i. fib = fib2, fib2 and fib1 are the previous two Fibonacci numbers
        ii. E.g., if fib, fib2, and fib1 were 34, 21, and 13, fib becomes 21, and fib2 and fib1
            become 13 and 8 respectively.
        iii. offset = index (Move the search space to the right part of the array)
4. if (fib2 > 0 AND element at offset+1 in A = E)
    a. Return that element is found at offset+1
5. Otherwise, output that element is not found
```

Implement the search algorithm by defining the function `fibonacciSearch` declared in `main.cpp`. A helper function called last2f_under is also provided. Complete the rest of the main function to accept the student's ID as input and search for the student. Output the index of the student if found, else output a message that the student was not found.

Task 2

Implement an insertion algorithm to insert a new student into the sorted array by defining the function `addStudent` declared in `main.cpp`.

Hint: You may reuse parts of the Fibonacci search algorithm to find the location to insert the student.



Bonus task 3

Implement an algorithm that can insert a new sorted list of students from a file called `newstudents.txt` into the array by defining the function `addStudents`.

Hint: Think of how Fibonacci or binary search can be modified to improve the efficiency of inserting a sorted list as compared to inserting them one by one.

Submit the following files in a zip file:

main.cpp

student.txt

newstudents.txt