Write your name and student id here: 

‘problems are not stop signs they are guidelines’ Robert H. Schuller

. Do not open this booklet until instructed.

- Please turn off your cell phones and put them in your bags.
- Manage your time. We start the exam at 9:30 and end the quiz at 10:00. You have 30 minutes.
- There are 4 pages (including this cover page and an extra blank page). Write your answers in the provided boxes.
- In the unlikely case that you find the exam too long/hard, do not panic. The marks will be scaled so that the highest mark gets the full mark.
- There are more important things in life than this course. Also, there are more important components to this course than this quiz. So, relax and have fun with algorithms.
Problem 1  [16 marks]  True/False questions:

- $n^2 \in O(n \log^3 n)$.  [False]
- $n(1 + \sin(n)) \in O(n)$.  [True]
- Stack is an Abstract Data Type which can be implemented with a data structure like an array.  [True]
- The following function has complexity $\max\{100 \log n, 4n \log n + \frac{1}{100} \log n, 2n + \log n\}$ which is $O(n \log n)$.  [False]

$$
\begin{align*}
  f(n) = & \begin{cases} 
    100 \log(n), & \text{for } 0 \leq n \leq 1397 \\
    4n \log n + \frac{1}{100} \log n, & \text{for } 1398 \leq n \leq 2019 \\
    2n + \log n, & \text{for } n > 2019 
  \end{cases}
\end{align*}
$$

Problem 2  [10 marks]

Analyze the following piece of pseudocode to give a bound on the running time as a function of $n$. Use big-Oh notation to summarize the final running time. Show your work.

```java
1. int joke = 0;
2. for ( int j = 1; j < n; j++ ) {
3.     for ( int i = 0; i < 2019; i++ ) {
4.         if (0 == 1) {
5.             joke ++;
6.             System.out.println("we love winter; we dislike warm weather");
7.         }
8.     }
9. }
```

Answer:  In the inner loop, one comparison is performed 2019 times (which returns false and the code never enters to the statements inside the if). So, the cost for the inner loop is $2019 \times 1 = 2019$. The outside loop is iterating $n$ times. So, the total cost for the for loops will be $2019n$. Adding the first statement, this sums up to $2019n + 1$ which is $O(n)$.

Problem 3  [20 marks]

A positive number is said to be good if either it is a prime or it can be written as the summation of two good numbers (e.g., 1 is not good, 2 and 3 are good since they are primes and 4 is good since it can be written as 2+2).

We are going to write a recursive code that indicates whether a given integer is good or not (assume the input is always positive). In the next page, you are given a public function `isGood(int x)` which initiates an array $F$ for memorization and calls a private recursive function `isGoodMemo(int x, int[] F)` that you have to implement. You are allowed to call function `public static boolean isPrime(int number)` which returns true if the input number is a prime number and false otherwise.
public static boolean isGood(int x)
{
    int[] F = new int[x+1];
    // F[i] = 0 indicates that F[i] is calculated and i is not a good number;
    // F[i] = 1 indicates that F[i] is calculated and i is a good number;
    // F[i] = -1 indicates that F[i] is not calculated

    for (int i=0; i<=x; i++)
        F[i] = -1; //at first, no value of F[i] is calculated
    F[x] = isGoodMemo(x, F); // call isGoodMemo to calculate F[x]
    if (F[x] == 1)
        return true;
    else
        return false;
}

// computes and returns an integer that indicates the value of F for index x
// output 1 indicates x is good (that is, x is prime or sum of two good numbers)
// output 0 indicates x is not good.
private static int isGoodMemo(int x, int[] F)
{
    // implement this!

    if (isPrime(x))
    {
        F[x] = 1;
        return 1;
    }

    for (int i = 1; i < x; i++)
    {
        if (F[i] == -1)
            F[i] = isGoodMemo(i, F);
        if (F[x-i] == -1)
            F[x-i] = isGoodMemo(x-i,F);
        if (F[i] == 1 && F[x-i] == 1)
        {
            F[x] = 1;
            return 1;
        }
    }
    F[x] = 0; // no good factorization found
    return 0;
}
Use this page only if you need more space!