Problem 1 (induction) (2 points)

Define:

\[ \text{SUM}(n) = \sum_{\emptyset \neq S \subseteq \{1,2,\ldots,n\}} \frac{1}{\text{product}(S)} \]

where \( \text{product}(S) \) is the product (multiplication) of all integers from \( S \). In other words \( \text{SUM}(n) \) is the sum of all terms \( \frac{1}{\text{product}(S)} \), over all possible nonempty subsets of \( \{1,2,\ldots,n\} \), for example:

\[ \text{SUM}(2) = \frac{1}{1} + \frac{1}{2} + \frac{1}{1} \]
\[ \text{SUM}(3) = \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{12} + \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{12} + \frac{1}{3} + \frac{1}{2} + \frac{1}{3} + \frac{1}{12} + \frac{1}{2} \]

(a) Give closed formula for \( \text{SUM}(n) \); (b) Show by induction correctness of the formula for \( \text{SUM}(n) \).

Problem 2 (analysis of simple algorithms) (2 points)

What is the value of \( s \) returned in each one of the functions below as a function of \( n \)? Use asymptotic notation to express this value in terms of \( n \). Write recurrence equations which relate \( A(n) \) to \( A(n-1) \) and \( B(n) \) to \( B(n-1) \).

```c
int A(int n) {
    int i,s; s=2;
    for(i=1;i<=n;i++) s += 5;
    return(s);
}

int B (int n) {
    int i,s; s=1;
    for(i=0;i<=n;i++) s=4*s+1;
    return(s);
}
```

Problem 3 (linear time algorithm) (2 points)

For the integer table \( A[0 \ldots n] \) we define the table \( \text{LEFT} \) as follows for \( 1 \leq i \leq n \):

\[ \text{LEFT}[i] = \max \{ k : (A[k] < A[i] \text{ and } 0 \leq k < i) \text{ or } (k = -1) \} \]

Write a program in C++ (as short as possible) which reads \( n + 1 \) numbers (the values of \( A[0] \), \( A[1] \), \( A[2] \) ... \( A[n] \)) and then computes in \( O(n) \) time (in total) and writes the values \( \text{LEFT}[1] \), \( \text{LEFT}[2] \) ... \( \text{LEFT}[n] \).

Problem 4 (asymptotics) (2 points)

Prove formally the following facts:

(a) \( n^3 + 10 = O((\frac{1}{10}n - 1)^3) \)

(b) \( 1 \times 2^1 + 2 \times 2^2 + 3 \times 2^3 + 4 \times 2^4 + \ldots n \times 2^n = O(n \times 2^n) \)