

1. (*Equivalence of Weak and Strong induction*) The definitions of weak and strong induction are given below:

(*Strong induction*) To prove that $P(n)$ is true for all positive integers, where $P(\cdot)$ is a propositional function, we complete two steps:

- Basis step: We show that $P(1)$ is true.
- Inductive step: We show that for all $k \geq 1$, if $P(1), P(2), \dots, P(k)$ are true, then $P(k+1)$ is true.

(*Weak induction*) To prove that $P(n)$ is true for all positive integers, where $P(\cdot)$ is a propositional function, we complete two steps:

- Basis step: We show that $P(1)$ is true.
- Inductive step: We show that for all $k \geq 1$, if $P(k)$ is true, then $P(k+1)$ is true.

Argue that any statement that can be proved using strong induction can also be proved using weak induction and vice versa.

2. (*Loop invariant*) Consider the following pseudocode:

```
IRoundedLog(x)
· i = 0; y = 1
· if(x ≤ 0) return("Error")
· while (y < x)
  · i = i + 1
  · y = 2 * y
· return(i)
```

- (a) What is the output of the algorithm for inputs 2, 3, 4, 5, 6, 7, 8, 9?
- (b) Argue that the above algorithm outputs $\lceil \log_2 x \rceil$ for any positive integer input x . Define an appropriate loop invariant. Argue that the loop invariant holds using mathematical induction. Use the fact that the loop invariant holds at the end of the while-loop to show that the algorithm outputs $\lceil \log_2 x \rceil$.
3. Answer the following:
- (a) State true or false: $2^{\sqrt{\log_2 n}}$ is $O(n)$.
- (b) Give reason for your answer to part (a).
4. Answer the following:
- (a) State true or false: 3^n is $O(2^n)$.
- (b) Give reason for your answer to part (a).
5. Consider functions $f(n) = 10n2^n + 3^n$ and $g(n) = n3^n$. Answer the following:
- (a) State true or false: $f(n)$ is $O(g(n))$.
- (b) State true or false: $f(n)$ is $\Omega(g(n))$.
- (c) Give reason for your answer to part (b).
6. Show using induction that for all $n \geq 0$, $1 + \frac{1}{2^1} + \frac{1}{2^2} + \frac{1}{2^3} + \dots + \frac{1}{2^n} = \frac{1 - (\frac{1}{2})^{n+1}}{1 - \frac{1}{2}}$.
7. Consider the following recursive function:

```

F(n)
- If ( $n > 1$ ) F( $n/2$ )
- Print("Hello World")

```

Let $R(n)$ denote the number of times this function prints "Hello World" given the positive integer n as input.

- (a) What is $R(n)$, in big-O notation as a function of n ?
 - (b) Give reason for your answer to part (a).
8. Consider the following recursive algorithm that is supposed to convert any positive integer in decimal to binary format. $\lfloor \cdot \rfloor$ denotes the floor function, $n\%2$ denotes the remainder when n is divided by 2, and $\parallel$ denotes concatenation.

```

RecDecimalToBinary(n)
- if( $n = 0$  or  $n = 1$ )return( $n$ )
-return(RecDecimalToBinary( $\lfloor n/2 \rfloor$ )  $\parallel$   $n\%2$ )

```

Prove that the above algorithm is correct. Proving the correctness of recursive programs such as the above is simple using mathematical induction. We simply induct on the input n . This means that the propositional function we shall work with is:

$P(n)$: The program is correct on input n , i.e., it correctly outputs the binary of the input number n .

Arguing the correctness of the program is the same as proving the statement $\forall n, P(n)$, which can be shown through induction. Complete this induction argument.

9. Show that:
- (a) If $d(n) = O(f(n))$ and $f(n) = O(g(n))$, then $d(n) = O(g(n))$.
 - (b) $\max\{f(n), g(n)\} = O(f(n) + g(n))$.
 - (c) If $a(n) = O(f(n))$ and $b(n) = O(g(n))$, then $a(n) + b(n) = O(f(n) + g(n))$.
10. Consider the two algorithms given below. In the input, A denotes an integer array and n denotes the size of the array. Analyse the running time of these algorithms and express the running time in big-O notation.

```

Alg1(A, n)
- for  $i = 1$  to  $n$ 
  -  $j \leftarrow i$ 
  - while( $j < n$ )
    -  $A[j] \leftarrow A[j] + 10$ 
    -  $j \leftarrow j + 3$ 

```

```
Alg2( $A, n$ )  
  - for  $i = 1$  to  $n$   
    - for  $j = 2i$  to  $n$   
      -  $A[i] \leftarrow A[j] + 1$ 
```

11. Consider the following problem:

ALL-ZEROS: Given the description $\langle A \rangle$ of an algorithm A , determine if this algorithm halts on all inputs with output 0.

An algorithm P is said to solve the above problem if $P(\langle A \rangle)$ halts and outputs 1 when A is an algorithm that halts on all inputs producing 0, and it halts and outputs 0 otherwise. Does there exist an algorithm P that solves the problem ALL-ZEROS?