

## Vocabulary

**Modus Ponens** **If P, then Q. P. Therefore, Q.** If the cake is made with sugar, then the cake is sweet. The cake is made with sugar. Therefore, the cake is sweet.

**Modus Tollens** **If P, then Q. Not Q. Therefore, not P.** If the cake is made with sugar, then the cake is sweet. The cake is not sweet. Therefore, the cake is not made with sugar.

**Onto Function** For every element Y in the codomain Y of F there is at least one element X in the domain X of X. Horizontal Line Test.

**One-To-One Function** A function for which every element of the range of the function corresponds to exactly one element of the domain.

**Bijection** Each element of one set is paired with exactly one element of the other set, and each element of the other set is paired with exactly one element of the first set.

## Vocabulary (cont)

**Domain** The set of all possible input x-values which will make the function "work", and output y-values.

**Codomain** The set of all possible output values of a function.

**Range** The set of **actual** output values of a function.

**Preimage** Another word for **Domain**

**Image** Another word for **Codomain**

**Fibonacci Sequence**  $F(n) = F(n-1) + F(n-2)$

**Universal Quantifier**  $\forall$  Expresses that the statements within its scope are true for everything, or every instance of a specific thing.

**Existential Quantifier**  $\exists$  Expresses that the statements within its scope are true for at least one instance of something.

**Scope** Denoted by symbols such as parenthesis and brackets to identify the section of the wff to which the quantifier applies.  $(\forall x)[P(x) \rightarrow Q(x)]$  - the scope of  $\forall x$  is found within the brackets.

## Vocabulary (cont)

**Universal Instantiation** Lets you remove  $\forall$  from a predicate.

**Existential Instantiation** Lets you remove  $\exists$  from a predicate. - Must be used before Universal Instantiation

**Method/Subroutine** A subroutine (such as a function) returns a value. A method is a subroutine or function that you can call on an object in an OO language.

**Principle of Well-Ordering** Every collection of positive integers that contains any members at all has a smallest number

**1st Principle of Mathematical Induction** Two assertions: 1. You can reach the first rung 2. Once you get to a rung, you can always climb to the next one up. (Implication). 1.  $P(1)$  2. For any positive integer  $k$ ,  $P(k) \rightarrow P(k+1)$

**2nd Principle of Mathematical Induction** Show that it's true for  $P(1)$ , assume it's true for some value "k", use that assumption to show that it's true for  $K+1$

**Binomial Theorem** Expands binomials.  $(a+b)^2 = a^2 + 2ab + b^2$

C

By **ajhalling**  
[cheatography.com/ajhalling/](http://cheatography.com/ajhalling/)

Not published yet.  
Last updated 25th October, 2019.  
Page 1 of 2.

Sponsored by **CrosswordCheats.com**  
Learn to solve cryptic crosswords!  
<http://crosswordcheats.com>

### Vocabulary (cont)

**Pascal's Triangle** a triangular array of numbers in which those at the ends of the rows are 1 and each of the others is the sum of the nearest two numbers in the row above (the apex, 1, being at the top).

**1st Order Recurrence Relation** No need to find the two that precede it. Does it fit the pattern?  $S(n) = cS(n-1) + g(n)$   
Step 1. Find C. Find g(n).  
Step 3. Plug n Chug.

**2nd Order Recurrence Relation** Requires the values from the previous two solutions. Fibonacci sequence is an example of 2nd Order RR.

**Closed-Form Solution** a mathematical expression that can be evaluated in a finite number of operations. No recurrence.

**Binary Predicate** Tests the truth value of a predicate which takes two arguments.

**Domain of Interpretation** Explains what is objects the predicate has meaning over. If  $P(x)$  x lives in the water, domain could be sea turtles.

### Vocabulary (cont)

**Big-O** Explains where the "bulk" of the work is happening in a function. Drops coefficients.

**Computational Complexity** The amount of resources required for running an algorithm

**Permutation** An ordered arrangement of objects. Multiply the factorial of the  $P(n,r)$  values to find the values.  $n!/n-r!$

**Factorial** A value multiplied by the value before it subsequently to 1.  $5! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1$

**Combination** An unordered arrangement of objects  $C(n,r)$ .  $n!/(r!(n-r)!)$

### Proofs Using Predicate Logic

#### Identify the Scope of a Variable in a Predicate

#### Identify the Scope of a Variable in a Program

#### Identify the Correct Negation of a Predicate

#### Big-O Value for the Complexity of an Algorithm

#### Do a Proof Using Mathematical Induction

#### Determine Whether a Relation is a Function or Not

#### Identify the Domain and Codomain of a Function

#### Classify a Function as Onto, 1-to-1, or Bijection

### Convert English Statements to Predicate Statements

### Expand a Binomial Using Binomial Theorem

### Write the First Few Rows of Pascal's Triangle

### Derive Closed Form Solution for 1st & 2nd Order RR

### Behavior of Java AND/OR Operators &, &&, |, ||