Personalized Study Guide

Of course. Here is a rigorous, detailed, and personalized study guide to help a student master the concepts of Boolean Expressions and Conditionals.

AP Computer Science Principles Study Guide: Mastering Conditionals

Hello! Welcome to your personalized study guide. You've shown that you're ready to tackle one of the most fundamental and powerful concepts in programming: **conditionals**. This guide is designed to break down everything you need to know, from the ground up, leaving you with a rock-solid understanding. Let's get started.

Topic 3.5: Boolean Expressions

1. Topic Overview: The Computer's Yes/No Questions

Think of a computer program as a very obedient but very simple-minded assistant. It can't guess what you want; you have to give it exact instructions. A huge part of those instructions involves making decisions. At the heart of every decision is a simple **yes/no question**.

Analogy: Imagine a traffic light. It asks a simple question: "Is it safe for cars to go?" If the answer is "yes," the light turns green. If the answer is "no," it turns red. In programming, these "yes/no" answers are called **Boole an values**: `true` (for yes) and `false` (for no). **Boolean expressions** are the questions we construct for the computer to ask, which always result in a `true` or `false` answer.

You use this logic every day. "Is it raining?" (a Boolean question). If `true`, you grab an umbrella. If `false`, you leave it at home. Mastering Boolean expressions is learning how to ask your computer the right questions so it can make smart decisions.

2. Deconstructing the Essential Knowledge

Let's break down the key ideas from the CED.

Essential Knowledge (EK) AAP-2.E.1 & AAP-2.E.2: Relational Operators - The Tools of Comparison

* **Elaborate and Explain:**
* **Boolean Value:** A data type that can only be one of two values: `true` or `false`.
* **Relational Operators:** These are the symbols we use to compare two values. They are the "verbs" in our Boolean questions. The AP Exam Reference Sheet provides the following:
* `a == b`: Is `a` **equal to** `b`?
* `a != b`: Is `a` **not equal to** `b`?
* `a > b`: Is `a` **greater than** `b`?
* `a < b`: Is `a` **less than** `b`?
* `a >= b`: Is `a` **greater than or equal to** `b`?

* `a <= b`: Is `a` **less than or equal to** `b`?

* **Provide Concrete Examples:**
* **Everyday Analogy:** Imagine you're at an amusement park ride with a height requirement. The sign says, "You must be at least 48 inches tall." The question the ride operator asks is: `yourHeight >= 48`. If your height is 50 inches, the expression `50 >= 48` evaluates to `true`. If your height is 45 inches, it evaluates to `false`.
* **Code Example:** Let's say we have a variable `score`. We want to know if the player has won by reaching 100 points.

```
score <- 100
hasWon <- (score == 100) // hasWon is now true

score <- 99
hasWon <- (score == 100) // hasWon is now false
```

* **Address Common Misconceptions:**
* **`==` vs. `<-`**: This is the most common mistake!
* `<-` (or `=` in many languages) is the **assignment operator**. It *gives* a variable a value. `myScore <- 10` means "put the value 10 into the variable `myScore`".
* `==` is the **equality operator**. It *compares* two values and results in `true` or `false`. `myScore == 10` asks, "is the value in `myScore` equal to 10?". You use `<-` to set a value and `==` inside a conditional to check it.

Essential Knowledge (EK) AAP-2.F.1 - AAP-2.F.5: Logical Operators - Combining Questions

* **Elaborate and Explain:**
* **Logical Operators:** Sometimes one question isn't enough. We need to combine questions. That's where logical operators come in. The AP Exam uses `AND`, `OR`, and `NOT`.
* `condition1 AND condition2`: Evaluates to `true` only if **both** `condition1` and `condition2` are `true`.
* `condition1 OR condition2`: Evaluates to `true` if **at least one** of the conditions is `true`. It's only `false` if both are `false`.
* `NOT condition`: **Inverts** the value of the condition. `NOT true` becomes `false`, and `NOT false` becomes `true`.

* **Provide Concrete Examples:**
* **Everyday Analogy:** To get a driver's license, you must be at least 16 years old AND have passed the driving test.
* `isOldEnough <- (age >= 16)`
* `passedTheTest <- true`
* `canGetLicense <- (isOldEnough AND passedTheTest)`
* If you are 17 (`isOldEnough` is `true`) but haven't passed the test (`passedTheTest` is `false`), then `canGetLicense` is `false` because `true AND false` is `false`.
* **Code Example:** A video game character can enter a special room if they have a key OR if their skill level is over 90.

```
hasKey <- false
skillLevel <- 95
canEnter <- (hasKey == true) OR (skillLevel > 90) // canEnter is true
```

Even though `hasKey` is false, `skillLevel > 90` is true, so the whole `OR` expression becomes `true`.

* **Address Common Misconceptions:**
* **How `OR` works:** Students often think `OR` means "one or the other, but not both." In logic, `OR` is inclusive. If both conditions are true, the `OR` expression is still `true`.
* **Order of Operations:** Logical operators have an order, just like `+` and `*`. The computer evaluates them in this order: `NOT`, then `AND`, then `OR`. Use parentheses `()` to control the order and make your code clearer. `(A OR B) AND C` is different from `A OR (B AND C)`.

3. Mastering the Learning Objectives

Learning Objective (LO) AAP-2.E & AAP-2.F: Write and evaluate expressions using relational and logical operators.

* **Actionable Guidance:** When you see a complex Boolean expression, evaluate it step-by-step:

1. **Parentheses First:** Evaluate anything inside `()` first, from the inside out.
2. **Relational Operators Next:** Solve all the comparisons (`>`, `==`, `<=`, etc.). This will turn parts of your expression into simple `true` or `false` values.
3. **Logical Operators Last:** Apply the logical operators in their order of precedence:
   * First, apply any `NOT`s.
   * Then, apply any `AND`s.
   * Finally, apply any `OR`s.

* **Illustrative Scenario:**
* **Problem:** A movie streaming service offers a family discount if a user is a "premium" member and has more than 3 profiles, OR if they are a "new" member. Let's write a Boolean expression for this.
* **Variables:** `memberType` (a string), `profileCount` (a number).
* **Thought Process:**

1. The first condition is for premium members: `(memberType == "premium") AND (profileCount > 3)`
2. The second condition is for new members: `(memberType == "new")`
3. The user gets a discount if one condition *OR* the other is met. So we combine them: `((memberType == "premium") AND (profileCount > 3)) OR (memberType == "new")`

* **Let's evaluate:**
* If `memberType` is "premium" and `profileCount` is 5: `(true AND true) OR false` -> `true OR false` -> `true`. **Discount!**
* If `memberType` is "premium" and `profileCount` is 2: `(true AND false) OR false` -> `false OR false` -> `false`. **No discount.**
* If `memberType` is "new" and `profileCount` is 1: `(false AND false) OR true` -> `false OR true` -> `true`. **Discount!**

4. Practice Makes Perfect

1. **Multiple-Choice:** The variables `onSale` (Boolean) and `quantity` (Number) are used to determine if an online order receives free shipping. Which expression evaluates to `true` if an item is NOT on sale and the quantity is 20 or more?
A) `onSale == true AND quantity <= 20`
B) `NOT onSale == false AND quantity > 20`
C) `NOT onSale == true OR quantity >= 20`
D) `(NOT onSale) AND (quantity >= 20)`

2. **Multiple-Choice:** Consider the following variables: `x <- 10`, `y <- 20`, `z <- 10`. What is the result of the

expression `(x == z) AND ((y > x) OR (z == y))`?
A) `true`
B) `false`
C) `10`
D) An error will occur.

3. **Short-Answer:** In your own words, explain the difference between the `AND` and `OR` logical operators. Provide a real-world example for each that is different from the ones in this guide.

4. **Short-Answer:** A video game has a quest that is available only to players who are level 25 or higher and are a member of the "Mages" guild. Write a single Boolean expression that would evaluate to `true` if a player is eligible for this quest. Use the variables `playerLevel` and `playerGuild`.

5. **Code Analysis Challenge:** A program determines if a student makes the honor roll. The variables are `gpa` (a number from 0.0 to 4.0) and `absences` (a number). The logic is: "A student makes the honor roll if their GPA is 3.5 or higher, as long as they have no more than 3 absences. However, even if their absences are too high, they can still make the honor roll if their GPA is a perfect 4.0."
   * **Part A:** Write a single, complex Boolean expression representing this logic.
   * **Part B:** Show how your expression evaluates for the following cases:
   * `gpa <- 3.6`, `absences <- 2`
   * `gpa <- 3.8`, `absences <- 4`
   * `gpa <- 4.0`, `absences <- 5`

Topic 3.6: Conditionals

1. Topic Overview: The Crossroads of Code

Once your program has an answer to a Boolean question (`true` or `false`), it needs to act on it. This is where **conditionals** come in. They are the crossroads in your program that direct the flow of execution down different paths.

Analogy: Think of a **"Choose Your Own Adventure" book**. You read a page, and it ends with a question: "To fight the dragon, turn to page 45. To sneak past it, turn to page 52." The choice you make (the *condition*) determines which set of instructions (which page) you follow next. A conditional statement (`IF`) is like this choice. You check a condition, and if it's `true`, you execute one block of code. If it's `false`, you might do something else, or nothing at all.

2. Deconstructing the Essential Knowledge

Essential Knowledge (EK) AAP-2.G.1 & AAP-2.H.1-3: Selection with IF and IF-ELSE

   * **Elaborate and Explain:**
   * **Selection:** This is the general term for deciding which parts of an algorithm to run based on a condition.
   * **Conditional Statement (`IF`)**: This is the primary tool for selection.
   * **`IF` statement:** The simplest form. "IF this condition is true, THEN execute these statements." If the

condition is false, the program just skips over that block of code and continues on.

* **`IF-ELSE` statement:** A more complete choice. "IF this condition is true, THEN execute Block A. OTHERWISE (ELSE), execute Block B." This guarantees that *one* of the two blocks will always run.

* **Provide Concrete Examples:**
* **Everyday Analogy:**
* **`IF`:** "IF my alarm is going off, I will hit the snooze button." (If it's not going off, you do nothing about the alarm).
* **`IF-ELSE`:** "IF it is a weekday, I will go to school. ELSE, I will sleep in." (You are always doing one or the other).
* **Code Example:**
* **`IF` Statement:** Give a 10-point bonus if score is over 90.

```
score <- 95
IF (score > 90)
{
  score <- score + 10  // This line runs. score becomes 105.
}
// Program continues...
```

* **`IF-ELSE` Statement:** Display a message based on a passing grade.

```
grade <- 55
IF (grade >= 60)
{
  DISPLAY("You passed!")
}
ELSE
{
  DISPLAY("You need to study more.") // This block runs.
}
```

* **Address Common Misconceptions:**
* **"Can both blocks in an IF-ELSE run?"** No, never. It's an either/or structure. The program evaluates the condition and chooses exactly one path.
* **"Does every IF need an ELSE?"** No. An `IF` statement is perfectly valid on its own. You only add an `ELSE` when you have a specific action to perform when the condition is `false`.

3. Mastering the Learning Objectives

Learning Objective (LO) AAP-2.G & AAP-2.H: Express algorithms using selection and determine the results.

* **Actionable Guidance:** To translate a real-world problem into a conditional statement:

1. **Find the "Question":** Identify the condition that needs to be checked. This will be your Boolean expression inside the `IF()`.
2. **Find the "Yes Action":** What should happen if the condition is `true`? This code goes inside the first `{}` block.
3. **Find the "No Action" (if any):** Is there something specific that must happen only if the condition is `false`? If so, this code goes in the `ELSE {}` block. If there's no specific "no" action, you don't need an `ELSE`.

* **Illustrative Scenario:**
* **Problem:** A website needs to charge for shipping. If the order total is $50.00 or more, shipping is free. Otherwise, it's $5.00. We need to calculate the `finalTotal`.
* **Variables:** `orderTotal`, `shippingCost`, `finalTotal`.
* **Thought Process:**

1. **The Question:** Is the `orderTotal` greater than or equal to 50? -> `orderTotal >= 50.00`
2. **The "Yes Action":** The `shippingCost` should be 0. -> `shippingCost <- 0`
3. **The "No Action":** The `shippingCost` should be 5. -> `shippingCost <- 5.00`

* **Code Implementation:**

```
shippingCost <- 0.00 // Initialize
IF (orderTotal >= 50.00)
{
  shippingCost <- 0.00
}
ELSE
{
  shippingCost <- 5.00
}
finalTotal <- orderTotal + shippingCost
```

4. Practice Makes Perfect

1. **Multiple-Choice:** What is displayed after the following code segment is run?

```
temp <- 75
isCloudy <- false
IF (temp > 80)
{
  DISPLAY("Beach day!")
}
ELSE
{
  DISPLAY("Park day!")
}
IF (isCloudy)
{
  DISPLAY("Bring a jacket.")
}
```

A) `Beach day!`
B) `Park day!`
C) `Park day! Bring a jacket.`
D) `Beach day! Bring a jacket.`

2. **Multiple-Choice:** A program is intended to display "Child" if `age` is less than 13, and "Teen" if `age` is 13 or greater. Which code segment accomplishes this?
A) `IF (age > 13) { DISPLAY("Teen") } ELSE { DISPLAY("Child") }`
B) `IF (age < 13) { DISPLAY("Teen") } ELSE { DISPLAY("Child") }`
C) `IF (age >= 13) { DISPLAY("Teen") } ELSE { DISPLAY("Child") }`

D) `IF (age <= 13) { DISPLAY("Child") } ELSE { DISPLAY("Teen") }`

3. **Short-Answer:** Explain a situation where you would use an `IF` statement without an `ELSE`. Why is the `ELSE` block not necessary in that case?

4. **Short-Answer:** A variable `isLoggedIn` is `true` if a user is logged in and `false` otherwise. Write a code segment that `DISPLAY`s "Welcome back!" if the user is logged in and `DISPLAY`s "Please log in." if they are not.

5. **Code Analysis Challenge:** The following code is intended to give a 10% discount if `isMember` is true OR if `purchaseTotal` is over 100. It contains an error.

```
purchaseTotal <- 120.00
isMember <- false
discount <- 0.0
IF (isMember == true)
{
  discount <- 0.10
}
IF (purchaseTotal > 100.00)
{
  discount <- 0.10
}
ELSE
{
  discount <- 0.0
}
finalPrice <- purchaseTotal * (1 - discount)
DISPLAY(finalPrice)
```

* **Part A:** What will this code display? Is it correct based on the intention?
* **Part B:** Explain the logic error. Why does it fail?
* **Part C:** Rewrite the conditional logic correctly using a single `IF-ELSE` statement and the `OR` operator.

Topic 3.7: Nested Conditionals

1. Topic Overview: Decisions Within Decisions

Sometimes, one decision leads to another. You decide to go out to eat. Now you face a new decision: what kind of food? After you decide on pizza, you have another decision: what toppings? **Nested Conditionals** are how we model this in code. They are simply conditional statements placed inside the code block of another conditional statement.

Analogy: Think of a **flowchart** for getting dressed. The first decision diamond asks, "Is it cold outside?" If you follow the "Yes" path, you hit another decision diamond: "Is it raining?" This second question is *nested* within the "it's cold" path. Your clothing choice depends on the outcome of both decisions.

2. Deconstructing the Essential Knowledge

Personalized Study Guide

Essential Knowledge (EK) AAP-2.I.1: Nesting Conditionals

* **Elaborate and Explain:**
* **Nested Conditional Statements:** An `IF` or `IF-ELSE` statement that is located inside the `true` block or the `false` block of another `IF-ELSE` statement. This allows for more complex, multi-layered decision-making.

* **Provide Concrete Examples:**
* **Everyday Analogy:** Deciding what to do on a Saturday.

1. `IF (haveHomework == true)`
   * Do homework.
2. `ELSE` (I don't have homework)
   * `IF (friendsAreFree == true)`
   * Hang out with friends.
   * `ELSE` (friends are busy)
   * Watch a movie.

Notice the second `IF-ELSE` only happens if the first condition (`haveHomework == true`) is `false`.

* **Code Example:** Ticket pricing at a museum. Adults are $20. Children (under 18) are $12. But if the child is also a local resident, they get a further discount to $10.

```
age <- 15
isLocal <- true
ticketPrice <- 0

IF (age < 18)
{
  // This is the child path
  IF (isLocal == true)
  {
    ticketPrice <- 10  // Nested condition is true
  }
  ELSE
  {
    ticketPrice <- 12
  }
}
ELSE
{
  // This is the adult path
  ticketPrice <- 20
}
DISPLAY(ticketPrice) // Displays 10
```

* **Address Common Misconceptions:**
* **Indentation vs. Logic:** Programmers use indentation to make nested code readable, but the computer only cares about the curly braces `{}`. Mismatched braces are a common source of syntax errors, while bad indentation just leads to confusion. Always make sure your braces correctly define which `IF` an `ELSE` belongs to.
* **Tracing the Path:** Don't try to evaluate all paths at once. Pick one set of inputs and trace the single path the program will take through the nested structure.

3. Mastering the Learning Objectives

Learning Objective (LO) AAP-2.I: Write and determine the result of nested conditional statements.

* **Actionable Guidance:** To write a nested conditional:

1. **Identify the "Outer" or Main Question:** What is the first, most important decision? This is your outer `IF-ELSE`.
2. **Code the Outer Paths:** Write the code for the `true` and `false` blocks of that main decision.
3. **Find the "Inner" Questions:** Look within each of those paths. Is there another decision that needs to be made *after* the first one is resolved? That's your nested conditional.
4. **Place the Inner Conditional:** Write the new `IF-ELSE` statement *inside* the appropriate block of the outer one.

* **Illustrative Scenario:**
* **Problem:** A package delivery service calculates a surcharge. For packages under 5 lbs, there is no surcharge. For packages 5 lbs or heavier, there is a $10 surcharge. Additionally, for these heavier packages, if the destination is "international", there's an extra $15 surcharge on top of the $10.
* **Variables:** `weight`, `destination`, `surcharge`.
* **Thought Process:**

1. **Outer Question:** Is the weight less than 5? -> `weight < 5`
2. **Outer Paths:**
   * If `true`: `surcharge <- 0`
   * If `false` (meaning weight is >= 5): This path needs more logic.
3. **Inner Question (within the false/heavy path):** Is the destination "international"? -> `destination == "international"`

* **Code Implementation:**

```
surcharge <- 0
IF (weight < 5)
{
  surcharge <- 0
}
ELSE
{
  // Package is 5lbs or heavier
  surcharge <- 10
  IF (destination == "international")
  {
    surcharge <- surcharge + 15
  }
}
```

4. Practice Makes Perfect

1. **Multiple-Choice:** What is displayed after the following code runs, if `x` is 5 and `y` is 10?

```
IF (x < 10)
{
  IF (y < 10)
  {
```

```
      DISPLAY("A")
    }
    ELSE
    {
      DISPLAY("B")
    }
  }
  ELSE
  {
    DISPLAY("C")
  }
```

A) `A`
B) `B`
C) `C`
D) `A` and `B`

2. **Multiple-Choice:** Under which condition will the following code display "Result: 3"?

```
  IF (a > b)
  {
    IF (a > c)
    {
      DISPLAY("Result: 1")
    }
    ELSE
    {
      DISPLAY("Result: 2")
    }
  }
  ELSE
  {
    DISPLAY("Result: 3")
  }
```

A) `a = 5, b = 3, c = 1`
B) `a = 5, b = 3, c = 6`
C) `a = 3, b = 5, c = 1`
D) `a = 5, b = 5, c = 5`

3. **Short-Answer:** In your own words, describe what a "nested conditional" is. Provide a real-world example of a two-level decision that would require one.

4. **Short-Answer:** Look at the ticket pricing example from the guide. How would you modify the code to give a new discount: local *adults* (age >= 18) pay only $15?

5. **AP-Style FRQ Challenge:** A program needs to assign a letter grade based on a numerical `score`.
   * 90 or above is an "A"
   * 80 to 89 is a "B"
   * 70 to 79 is a "C"
   * 60 to 69 is a "D"
   * Below 60 is an "F"

Write a block of code using nested `IF-ELSE` statements that will assign the correct letter to the `letterGrade` variable based on the `score` variable. (Hint: A common pattern is `IF... ELSE IF... ELSE IF...`).

Answer Key and Solution Walkthroughs

<details>
<summary>Click to see answers and explanations</summary>

Topic 3.5: Boolean Expressions

1. **D)** `(NOT onSale)` is `true` when `onSale` is `false`. `(quantity >= 20)` checks the other condition. `AND` requires both to be true.
2. **A)** `(x == z)` is `(10 == 10)`, which is `true`. `(y > x)` is `(20 > 10)`, which is `true`. `(z == y)` is `(10 == 20)`, which is `false`. The inner expression becomes `(true OR false)`, which is `true`. The final expression is `true AND true`, which is `true`.
3. **Short-Answer:** `AND` requires all parts of the expression to be true for the whole thing to be true (e.g., "To board the plane, you need a ticket AND a photo ID"). `OR` requires only one part to be true for the whole thing to be true (e.g., "For payment, we accept a credit card OR cash").
4. **Short-Answer:** `(playerLevel >= 25) AND (playerGuild == "Mages")`
5. **Code Analysis Challenge:**
   * **Part A:** `((gpa >= 3.5) AND (absences <= 3)) OR (gpa == 4.0)`
   * **Part B:**
   * `gpa <- 3.6`, `absences <- 2`: `((true) AND (true)) OR (false)` -> `true OR false` -> `true`. (Honor Roll)
   * `gpa <- 3.8`, `absences <- 4`: `((true) AND (false)) OR (false)` -> `false OR false` -> `false`. (No Honor Roll)
   * `gpa <- 4.0`, `absences <- 5`: `((false) AND (false)) OR (true)` -> `false OR true` -> `true`. (Honor Roll)

Topic 3.6: Conditionals

1. **B)** The first `IF (temp > 80)` is false (75 is not > 80), so its `ELSE` block runs, displaying "Park day!". The second `IF (isCloudy)` is false, so its block is skipped.
2. **C)** This correctly checks if `age` is 13 or greater. If so, it displays "Teen". In all other cases (age is less than 13), it displays "Child".
3. **Short-Answer:** You would use an `IF` without an `ELSE` if you only need to take an action when a condition is true, and do nothing special if it's false. For example, applying a bonus to a score: `IF (playerScore > 1000) { bonusPoints <- 50 }`. If the score isn't over 1000, you just move on.
4. **Short-Answer:**

```
IF (isLoggedIn == true)
{
  DISPLAY("Welcome back!")
}
ELSE
{
  DISPLAY("Please log in.")
}
```

5. **Code Analysis Challenge:**
   * **Part A:** It will display `120.0`. The intention is for a 10% discount, making the price `108.0`. So, the code is incorrect.
   * **Part B:** The error is in using two separate `IF` statements. When `isMember` is false and

`purchaseTotal` is 120, the first `IF` is skipped. The second `IF` condition (`120 > 100`) is true, so `discount` is set to 0.10. However, this `IF` has an `ELSE` attached. Since the `IF` was true, the `ELSE` is skipped. This seems right, but consider if `isMember` was true and `purchaseTotal` was 50. The first `IF` sets `discount` to 0.10. Then the second `IF` (`50 > 100`) is false, so its `ELSE` block runs, resetting `discount` back to 0.0! The structure is flawed.

* **Part C:**

```
IF ((isMember == true) OR (purchaseTotal > 100.00))
{
  discount <- 0.10
}
ELSE
{
  discount <- 0.0
}
finalPrice <- purchaseTotal * (1 - discount)
DISPLAY(finalPrice)
```

Topic 3.7: Nested Conditionals

1. **B)** `x < 10` is true, so we enter the first block. Inside, `y < 10` is false (10 is not less than 10), so the inner `ELSE` runs, displaying "B".
2. **C)** We need the outer `ELSE` block to run, which means the condition `a > b` must be false. In choice C, `3 > 5` is false.
3. **Short-Answer:** A nested conditional is an IF statement inside another IF statement. It's used for multi-level decisions. Example: At a restaurant, the first decision is "Do you want a drink?" (IF). If yes, the nested decision is "Do you want soda or water?" (nested IF-ELSE).
4. **Short-Answer:** You would modify the `ELSE` block for adults.

```
ELSE // Adult path
{
  IF (isLocal == true)
  {
    ticketPrice <- 15
  }
  ELSE
  {
    ticketPrice <- 20
  }
}
```

5. **AP-Style FRQ Challenge:**

```
letterGrade <- ""
IF (score >= 90)
{
  letterGrade <- "A"
}
ELSE
{
  IF (score >= 80)
  {
```

```
        letterGrade <- "B"
      }
      ELSE
      {
        IF (score >= 70)
        {
          letterGrade <- "C"
        }
        ELSE
        {
          IF (score >= 60)
          {
            letterGrade <- "D"
          }
          ELSE
          {
            letterGrade <- "F"
          }
        }
      }
    }
```

</details>