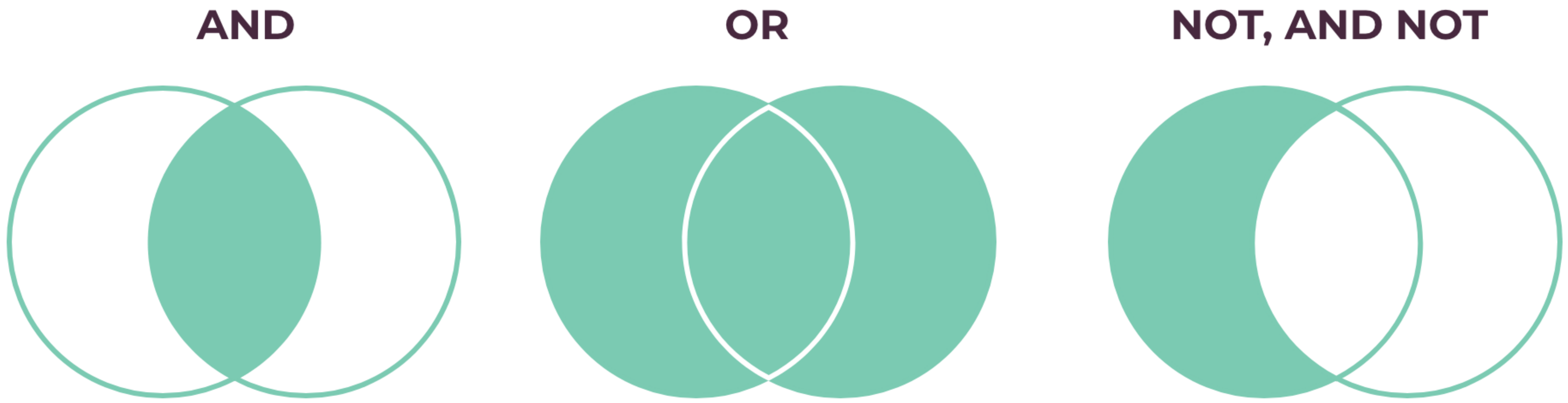
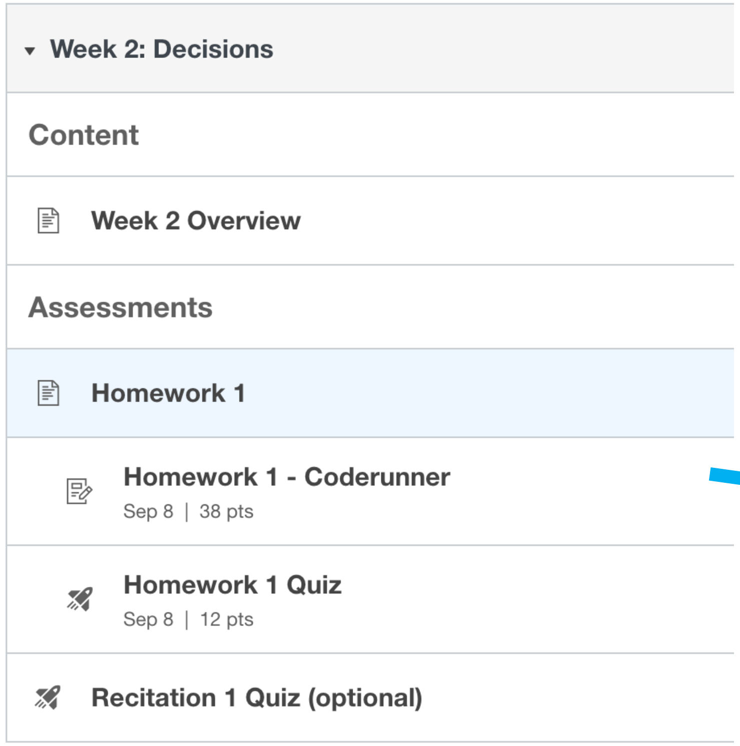


Boolean Operators

Due this week

- **Homework 1**
  - Write solutions in VS Code
  - Paste in Autograder, **Homework 1 CodeRunner**.
  - Complete the quiz
- Check the due date! **No late submissions!!**

Homework 1 - CodeRunner



▾ Week 2: Decisions

Content

Week 2 Overview

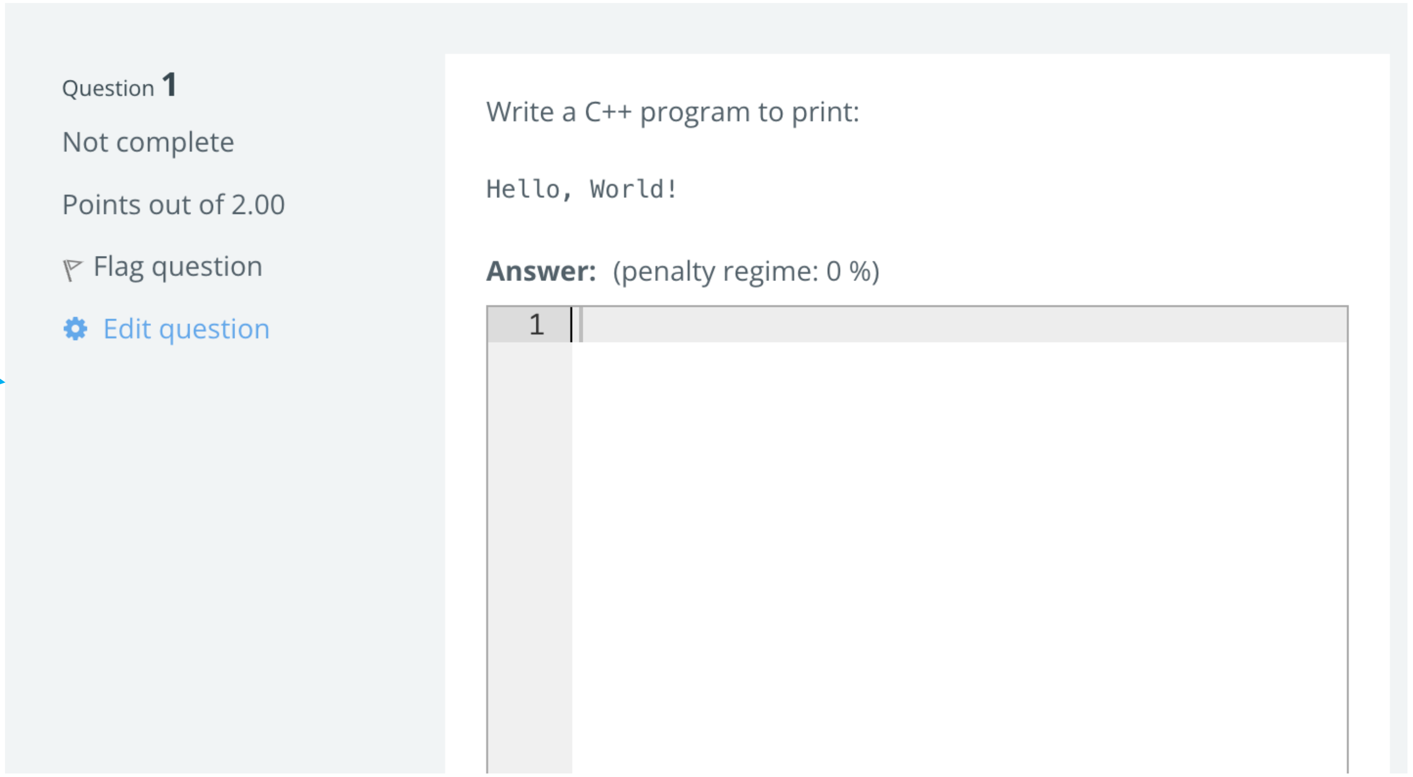
Assessments

Homework 1

Homework 1 - Coderunner
Sep 8 | 38 pts

Homework 1 Quiz
Sep 8 | 12 pts

Recitation 1 Quiz (optional)



Question 1
Not complete
Points out of 2.00
Flag question
Edit question

Write a C++ program to print:

```
Hello, World!
```

Answer: (penalty regime: 0 %)

Today

- Boolean Operators
- The `if else if` statement

Boolean Operators

Logical Operators

- **Example:** you need to write a program to process temperature values, and tests whether a given temperature corresponds to liquid water or to solid ice.
- At sea level, water freezes at 0 degrees Celsius and boils at 100 degrees Celsius.
- Water is liquid IF the temperature is greater than 0 AND less than 100

Logical Operators: And &&

- **Example:** you need to write a program to process temperature values, and tests whether a given temperature corresponds to liquid water or to solid ice.
- At sea level, water freezes at 0 degrees Celsius and boils at 100 degrees Celsius.
- Water is liquid IF the temperature is greater than 0 AND less than 100
- In C++, the && operator (called “and”) yields true only when both conditions that it joins are true:

```
if (temp > 0 && temp < 100)
{
      cout << “Liquid” << endl;
}
```

Truth Tables

- **Definition:** A truth table displays the value of a Boolean operator expression for all possible combinations of its constituent expressions.
- (You'll look at truth tables a lot more in CSCI 2824 (Discrete))
- So if A and B denote bool variables or Boolean expressions, we have:

A	B	A && B
true	true	true
true	false	false
false	true	false
false	false	false

A	B	A || B
true	true	true
true	false	true
false	true	true
false	false	false

A	!A
true	false
false	true

Logical Operators: And &&

```
if (temp > 0 && temp < 100)
{
      cout << “Liquid” << endl;
}
else
{
      cout < “Not liquid” << endl;
}
```

- If temp is within the 0 to 100 range, then both the left-hand side and right-hand side are true, so the whole expression in parens ( ) has value = true
- In all other cases, the whole expression’s value is false

Logical Operators: Or ||

- The || operator (called or) yields the result true if at least one of the conditions connected by it is true
- Written as two adjacent vertical bar symbols (above the Enter key)

```
if (temp <= 0 || temp >= 100)
{
     cout < “Not liquid” << endl;
}
```

- If either of the left-hand or right-hand side expressions is true, then the whole expression has value true
- **Question:** What is the only case in which “Not liquid” would appear?

Logical Operators: Not !

- Sometimes, you need to invert a condition with the logical not operator: !
- The ! operator takes a single condition and evaluates to true if the condition is false, and to false if the condition is true

```
if (!frozen)
{
      cout < “Not frozen” << endl;
}
```

- “Not frozen” will be written only when frozen contains the value false
- **Question:** What is the value of !false ?

Examples

- 0 < 200 && 200 < 100
- 0 < 200 || 200 < 100
- 0 < 200 || 100 < 200
- 0 < 200 < 100
- !(0 < 200)
- -10 && 10 > 0
- 0 < x && x < 100 || x == -1
- (!0 < x && x < 100) || x == -1

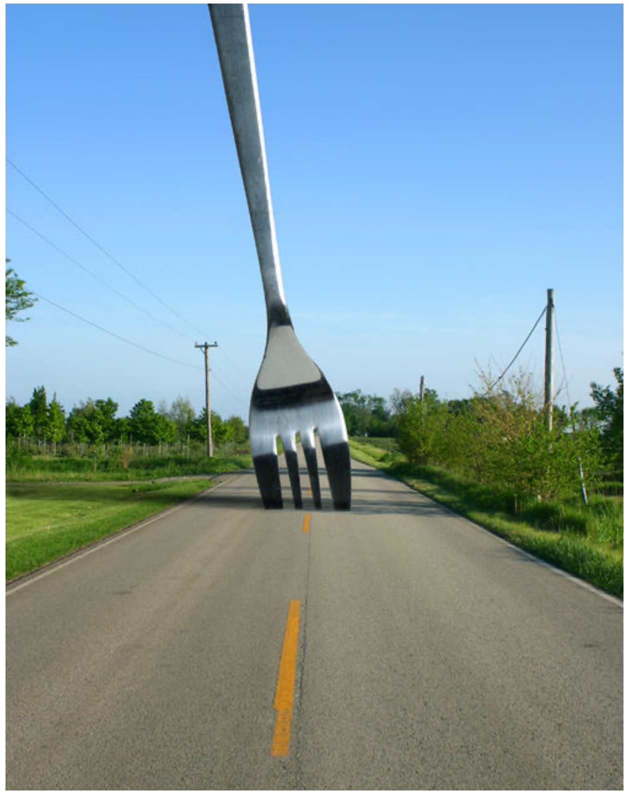
The `if` statement

How do you know that class has ended?

The `if` Statement

- The **`if`** statement is used to implement a decision
  - When a condition is fulfilled,
    one set of statements is executed
  - Otherwise,
    another set of statements is executed

- Like a fork in the road



Syntax of the `if()` Statement

```
if (condition)//never put a semicolon after the parentheses!!
{
   statement1; //executed if condition is true
}
else   //the else part is optional
{
   statement2; //executed if condition false
} //braces are optional but recommended
```

Common Error – The Do-nothing Statement

- This is *not* a compiler error.
- The compiler does not complain.
- It interprets this **`if`** statement as follows:
  - If floor is greater than 13, execute the do-nothing statement (semicolon by itself is the do-nothing statement)
  - Then execute the code enclosed in the braces.
- Any statements enclosed in the braces are no longer a part of the if statement.

```
if (floor > 13); // ERROR?
{
   floor--;
}
```

The `if` Statement: Elevator Example

We must write the code to control the elevator.

How can we skip the 13th floor?

`if()` Elevator Example Code

- If the user inputs 20, the program must set the actual floor to 19.
- Otherwise, we simply use the supplied floor number.

We need to decrement the input only under a certain condition:

`if()` Elevator Example Code

```
int floor;
cout << "Enter the desired floor: ";
cin >> floor;
int actual_floor;
if (floor > 13)   //never put a semicolon after the parentheses!!
{
   actual_floor = floor - 1; //
}
else
{
   actual_floor = floor;
}
```

Is the **else** part necessary?

`if()` Elevator Example without `else`

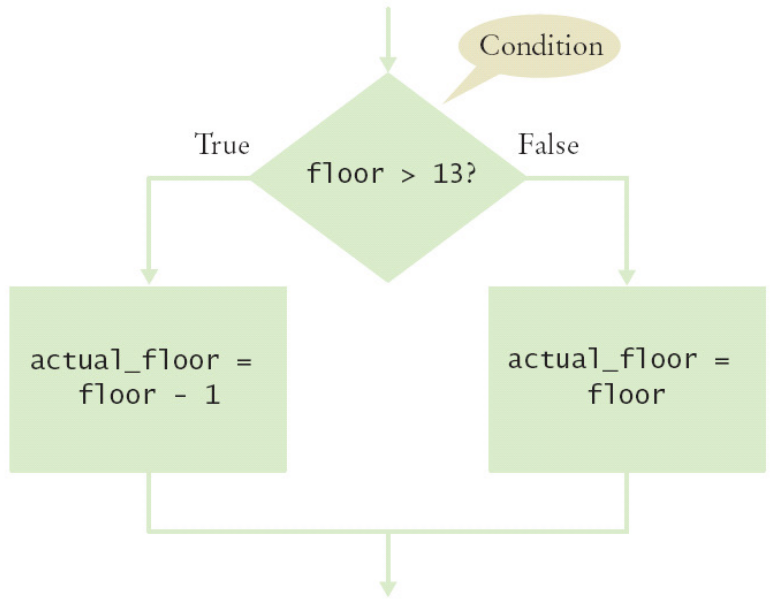
Here is another way to write this code:
We only need to decrement when the floor is greater than 13.

We can set **actual_floor** before testing:

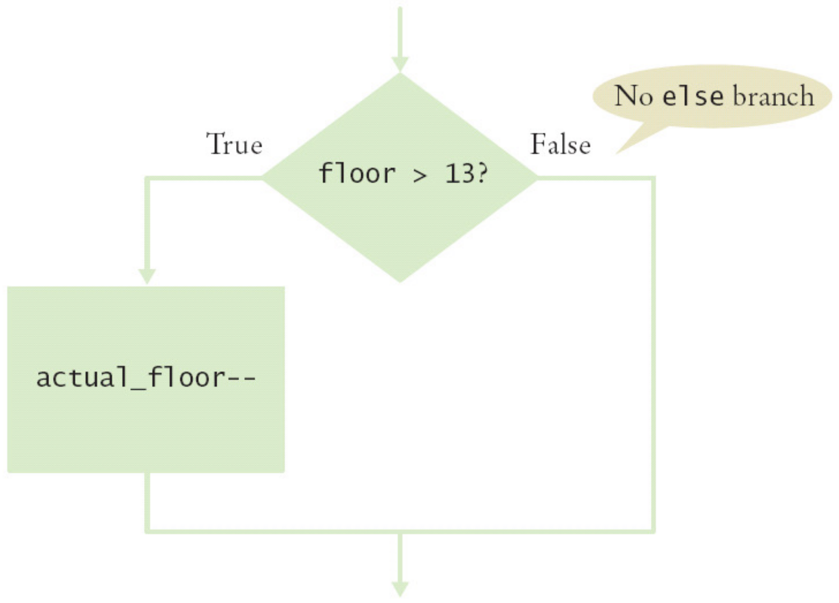
```
int actual_floor = floor;
if (floor > 13)
{
    actual_floor--;
} // No else needed
```

(And you'll notice we used the decrement operator this time.)

The `if` Statement Flowcharts



With else



Without else

The `if` Statement – Always use Braces

- When the body of an **`if`** statement consists of a single statement, you need not use braces:

```
if (floor > 13)
   floor--;
```

- However, it is a good idea to always include the braces:
  - the braces makes your code easier to read, and
  - you are less likely to make errors

The `if` Statement – Brace Layout

- Making your code easy to read is good practice.
- Lining up braces vertically helps.

```
if (floor > 13)
{
   floor--;
}
```

The `if` Statement – Removing Duplication

```
if (floor > 13)
{
     actual_floor = floor - 1;
     cout << "Actual floor: " << actual_floor << endl;
}
else
{
     actual_floor = floor;
     cout << "Actual floor: " << actual_floor << endl;
}
```

- Do you find anything redundant in this code?

The `if` Statement – Removing Duplication

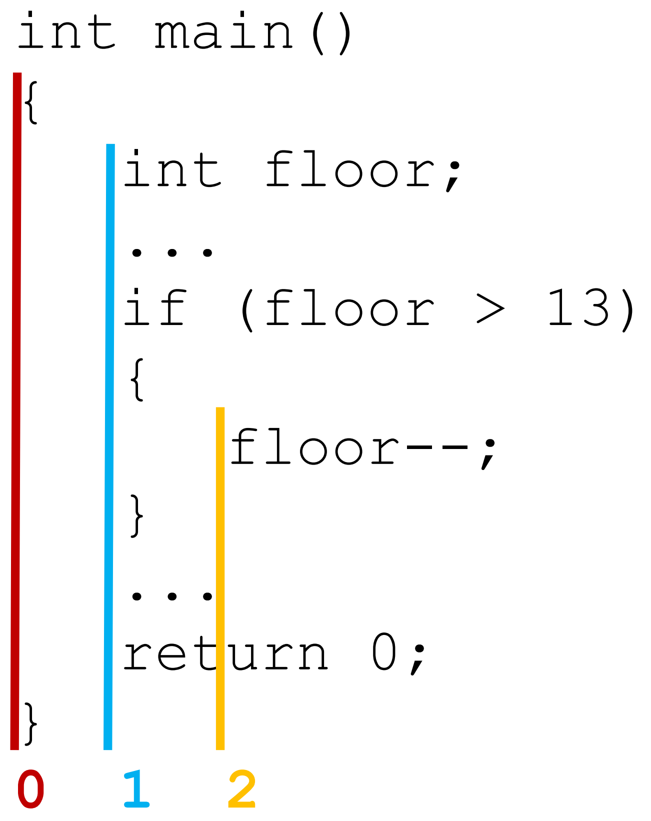
```
if (floor > 13)
{
     actual_floor = floor - 1;
}
else
{
     actual_floor = floor;
}
cout << "Actual floor: " << actual_floor << endl;
```

You can remove the duplication by moving the two identical cout statements outside of and after the braces, and of course deleting one of the two.

Nested Branches

The `if` Statement – Indent when Nesting

Block-structured code has the property that *nested* statements are indented by one or more levels.



```
int main()
{
    int floor;
    ...
    if (floor > 13)
    {
        floor--;
    }
    ...
    return 0;
}
```

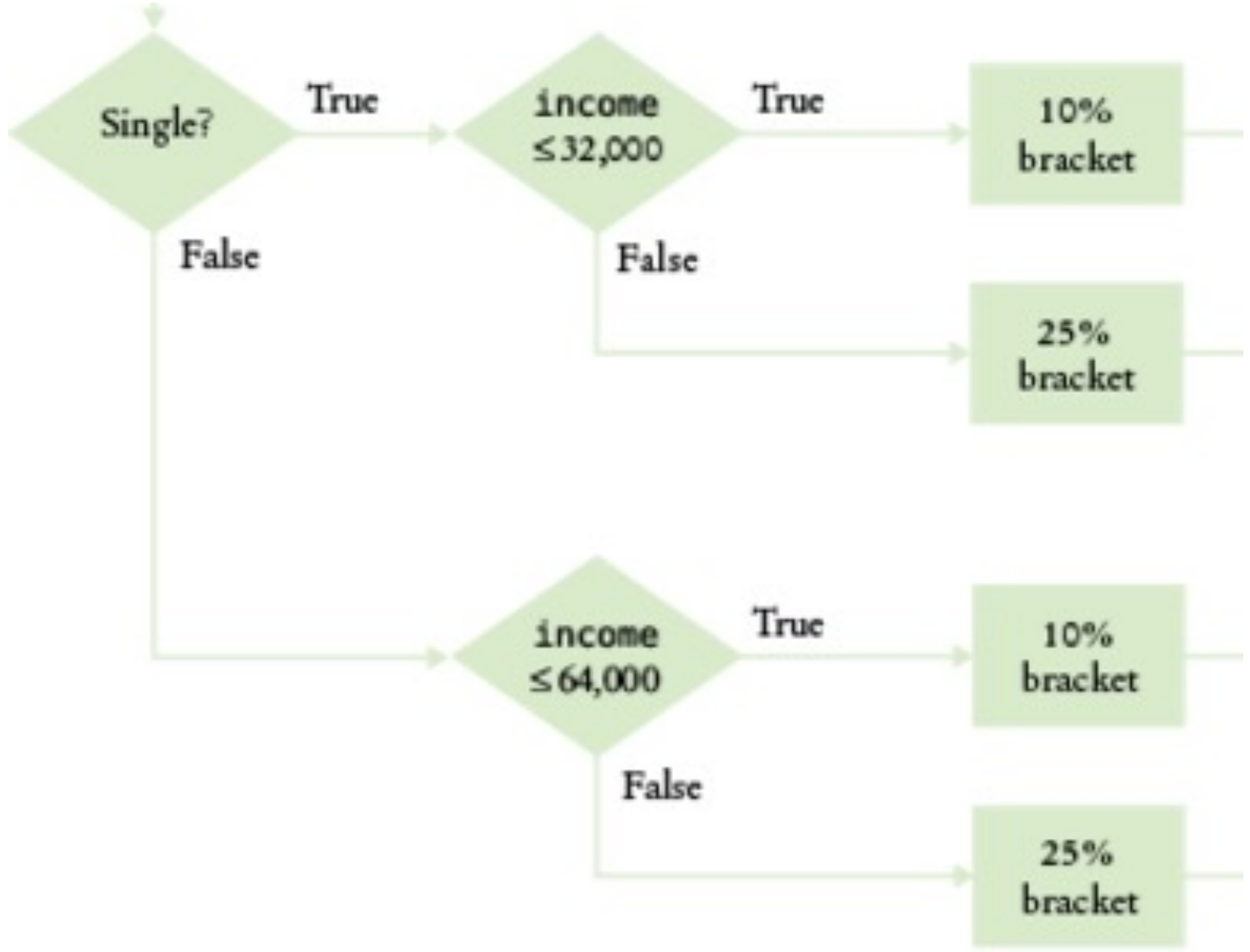
Indentation level

Nested Branches – Taxes

Table 4 Federal Tax Rate Schedule

If your status is Single and if the taxable income is	the tax is	of the amount over
at most $32,000	10%	$0
over $32,000	$3,200 + 25%	$32,000
If your status is Married and if the taxable income is	the tax is	of the amount over
at most $64,000	10%	$0
over $64,000	$6,400 + 25%	$64,000

In the United States different tax rates are used depending on the taxpayer's marital status – single rates are higher. Married taxpayers add their income together and pay taxes on the total. See the IRS table below from a recent year:



Flowchart for Tax Table Decisions

Nested Branches – Taxes – Complete Code part 1

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
   const double RATE1 = 0.10;
   const double RATE2 = 0.25;
   const double RATE1_SINGLE_LIMIT = 32000;
   const double RATE1_MARRIED_LIMIT = 64000;

   double tax1 = 0;
   double tax2 = 0;

   double income;
   cout << "Please enter your income: ";
   cin >> income;

   cout << "Please enter s for single, m for married: ";
   string marital_status;
   cin >> marital_status;
```

Nested Branches – Taxes – Complete Code part 2

```
if (marital_status == "s")
   {
      if (income <= RATE1_SINGLE_LIMIT)
      {
         tax1 = RATE1 * income;
      }
      else
      {
         tax1 = RATE1 * RATE1_SINGLE_LIMIT;
         tax2 = RATE2 * (income - RATE1_SINGLE_LIMIT);
      }
   }
else
```

Nested Branches – Taxes – Complete Code part 2

```
   {
         if (income <= RATE1_MARRIED_LIMIT)
         {
            tax1 = RATE1 * income;
         }
         else
         {
            tax1 = RATE1 * RATE1_MARRIED_LIMIT;
            tax2 = RATE2 * (income - RATE1_MARRIED_LIMIT);
         }
      }

      double total_tax = tax1 + tax2;

      cout << "The tax is $" << total_tax << endl;
      return 0;
  }
```