Solution

Date of MSE-1 : 19/09/24

Q1 Differentiate between compiler and interpreter (at least 4 differences).

Q2 What will be the output for the following code snippet:

```
int main( ) {
int a = 10, b = 0, c = 5 ;
if ( (a > 5 && + + b) ||  ( c= =4 && ++b)) {
cout<<" Inside If: a = " <<a<< " b =" <<b<<endl;
}
else{
cout << "Inside else: a =" << a << ", b=" << b <<endl;
}
return 0;
}
```

Q3 What goes behind the scene when you attempt to get an output from source code in C++? Elaborate the process steps with the help of a diagram.

04 Write a program to swap two positive integer numbers using bitwise operators. Provision to accept positive integer numbers should be made available.

Q5 Explain implicit and explicit type conversions by using any scenario. [Don't miss to explain the scenario.)

Q6 Design a menu driven code that does the following:
If '1' is entered, must be able to find the greatest among three positive integer numbers entered by the user.
If '2' is entered, must be able to find the square-root of a positive number entered by the user.
If any other integer is entered, code must be able to terminate with a suitable message.
[Don't miss to draw flowchart(s).]

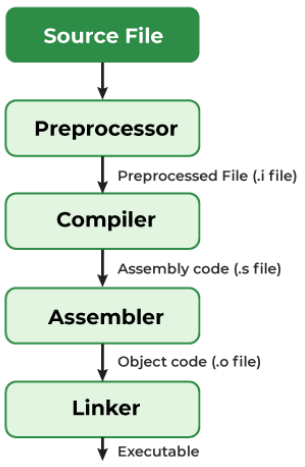
Q1. Differentiate between compiler and interpreter (at least 4 differences).

Feature	Compiler	Interpreter
Definition	Translates the entire source code into machine code before execution.	Translates and executes the source code line by line.
Output	Generates an intermediate object code or executable file.	Does not produce a separate output file; executes code directly.
Execution Speed	Generally faster after compilation, as the code is already translated.	Generally slower, as translation occurs during execution.
Error Detection	Detects errors after the entire code is compiled; errors are reported after compilation.	Detects errors line by line during execution; stops at the first error encountered.
Memory Usage	Requires more memory due to storage of the entire compiled code.	Generally uses less memory as it processes code one line at a time.
Examples	C, C++, Java (initial compilation step)	Python, Ruby, JavaScript
Development Cycle	Typically requires a longer compile-link-execute cycle.	More suitable for interactive programming and rapid prototyping.
Debugging	Debugging is more difficult since it requires recompilation for testing.	Easier to debug since it executes line by line.

Q2. What will be the output for the following code snippet:

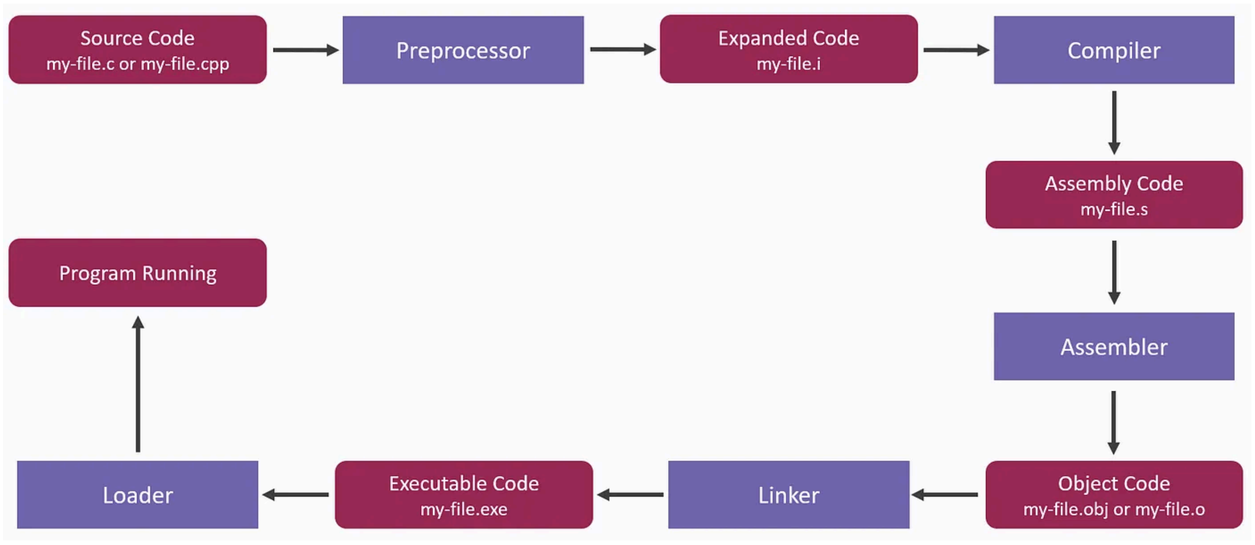
Inside If: a = 10 b =1

Q3. What goes behind the scene when you attempt to get an output from source code in C++? Elaborate the process steps with the help of a diagram.



- **Step 1:** Preprocessor converts Source code to Expanded code. Removal of Comments. The preprocessed output is stored in the **filename.i.**

- **Step 2:** Compiler converts the Expanded code to Assembly code. This file is in assembly-level instructions. It compiles filename.i and produces an; intermediate compiled output file **filename.s.**
- **Step 3:** Assembler converts the Assembly code to Object code. In this phase the **filename.s** is taken as input and turned into **filename.o** by the assembler. This file contains machine-level instructions.
- **Step 4:** Linker converts the Object code to Executable code. This is the final phase in which all the linking of function calls with their definitions is done. In this step, the tool known as **Linker** converts the Object code to Executable code which is the executable file **filename.exe**.
- **Step 5:** Finally, the Loader loads the executable file into memory.



C/C++ program execution flow

Q4. Write a program to swap two positive integer numbers using bitwise operators. Provision to accept positive integer numbers should be made available.

```
#include <iostream>
using namespace std;

int main() {
  int a, b;

  // Accepting positive integer inputs
```

```
    cout << "Enter the first positive integer: ";
    cin >> a;
    cout << "Enter the second positive integer: ";
    cin >> b;

    // Check if both numbers are positive
    if (a <= 0 || b <= 0) {
        cout << "Please enter positive integers only." << endl;
        return 1; // Exit the program with an error code
    }

    // Swapping using bitwise XOR
    a = a ^ b; // Step 1: a now holds the XOR of a and b
    b = a ^ b; // Step 2: b now holds the original value of a
    a = a ^ b; // Step 3: a now holds the original value of b

    cout << "After swapping:" << endl;
    cout << "First integer: " << a << endl;
    cout << "Second integer: " << b << endl;
    return 0;
}
```

Q5. Explain implicit and explicit type conversions by using any scenario. [Don't miss to explain the scenario.)

Implicit Conversion: Automatically converts types but may lead to inaccuracies.
Explicit Conversion: Programmer specifies the type conversion, ensuring accuracy

Implicit Example:
Scenario: Temperature Conversion
Suppose we want to convert a temperature from Celsius to Fahrenheit. We have an integer representing the Celsius temperature and we want to calculate the Fahrenheit temperature using the formula: **Fahrenheit=(Celsius×59)+32**

Program: Implicit Conversion

```
#include <iostream>
using namespace std;
int main() {
    int celsius = 25; // Celsius temperature as an integer
```

```
  // Implicit conversion occurs here
  double fahrenheit = (celsius * 9 / 5) + 32; // Integer division
  cout << "Fahrenheit (Implicit): " << fahrenheit << endl; // Outputs 77
  return 0;
}
```

Explanation:

- In the expression `(celsius * 9 / 5)`, both `celsius` and `9` are integers. The multiplication results in an integer, and the division by `5` also performs integer division, resulting in `15` (the decimal part is discarded).
- When adding `32`, the result is `47`, which is then implicitly converted to a `double` when assigned to `fahrenheit`. The calculation is incorrect because it does not reflect the expected Fahrenheit value.

Explicit Type Conversion:
Program:

```
#include <iostream>
using namespace std;

int main() {
double pi = 3.14159;
int approx_pi = (int)pi;  // Explicitly converts double to int
cout<< approx_pi; // Output will be 3
  }
```

Explanation:

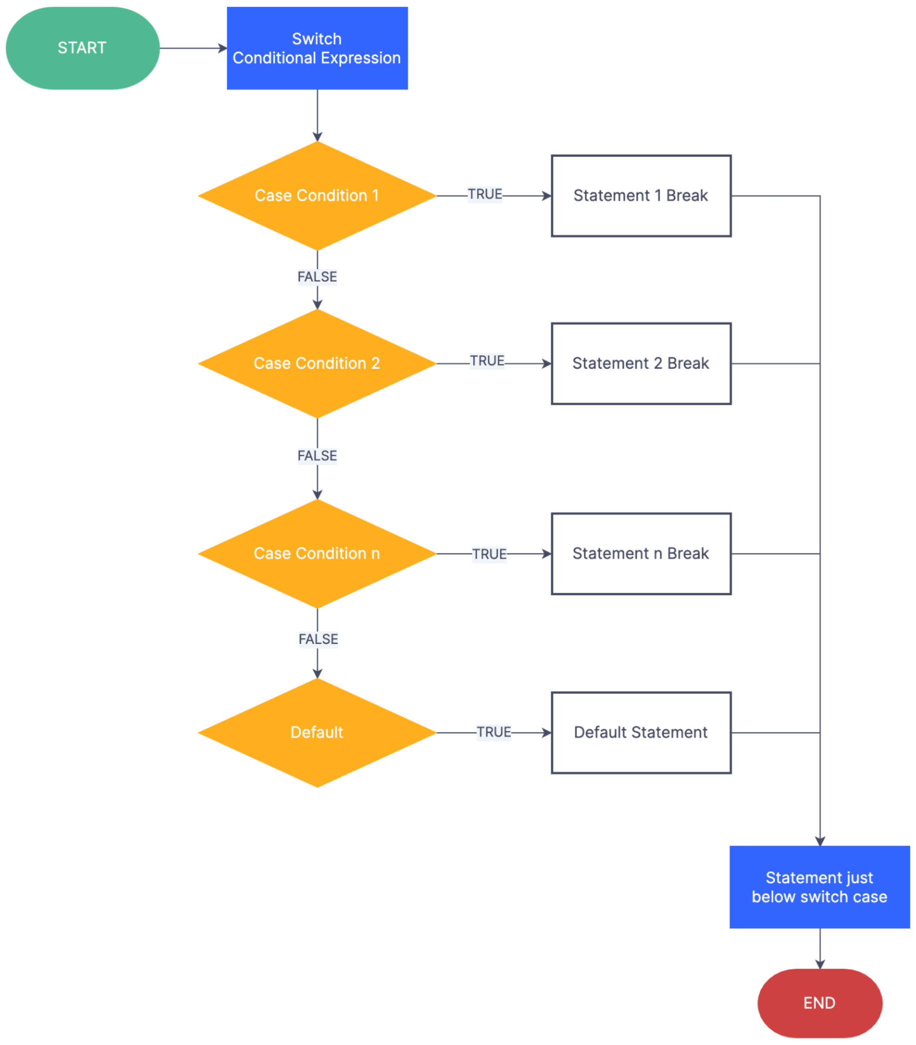
- double pi = 3.14159; This line declares a variable pi of type double and assigns it the value 3.14159.
- **int approx_pi = (int)pi;**Here, we use ‘(int)pi’ to explicitly convert `pi` to `int`. The (int) before pi is a **type cast**, which forces the conversion from one data type to another. In this case, it converts the double value of 3.14159 to an int.
- When converting a floating-point number to an integer, the fractional part (anything after the decimal point) is truncated, **not rounded**. So, 3.14159 becomes 3.
- **cout << approx_pi;**This line outputs the value of approx_pi to the console.

This conversion is called **explicit type conversion** or **type casting**, where the programmer is explicitly telling the compiler to convert the type.

Q6. Design a menu driven code that does the following:
If '1' is entered, must be able to find the greatest among three positive integer numbers entered by the user.
If '2' is entered, must be able to find the square-root of a positive number entered by the user.
If any other integer is entered, code must be able to terminate with a suitable message.
[Don't miss to draw flowchart(s).]

```
#include <iostream>
#include <cmath>  // For sqrt function
using namespace std;

int main() {
    int choice;
    cout << "Menu: \n";
    cout << "1. Find the greatest among three positive integers.\n";
    cout << "2. Find the square root of a positive number.\n";
    cout << "Enter your choice (1 or 2): ";
    cin >> choice;

    if (choice == 1) {
      // Find the greatest among three positive integers
        int a, b, c;
        cout << "Enter three positive integers: ";
        cin >> a >> b >> c;

        if (a > 0 && b > 0 && c > 0) {
            int greatest = a;
            if (b > greatest) greatest = b;
            if (c > greatest) greatest = c;
            cout << "The greatest number is: " << greatest << endl;
        } else {
            cout << "Please enter only positive integers." << endl;
        }
    } else if (choice == 2) {
      // Find the square root of a positive number
        double num;
        cout << "Enter a positive number: ";
        cin >> num;

        if (num > 0) {
            cout << "The square root of " << num << " is: " << sqrt(num) << endl;
        } else {
            cout << "Please enter a positive number." << endl;
        }
    } else {
        // Terminate the program with a message
        cout << "Exiting the program. Invalid choice entered." << endl;
    }

    return 0;
}
```

Switch Case Flowchart


START
Switch Conditional Expression
Case Condition 1
TRUE
Statement 1 Break
FALSE
Case Condition 2
TRUE
Statement 2 Break
FALSE
Case Condition n
TRUE
Statement n Break
FALSE
Default
TRUE
Default Statement
Statement just below switch case
END