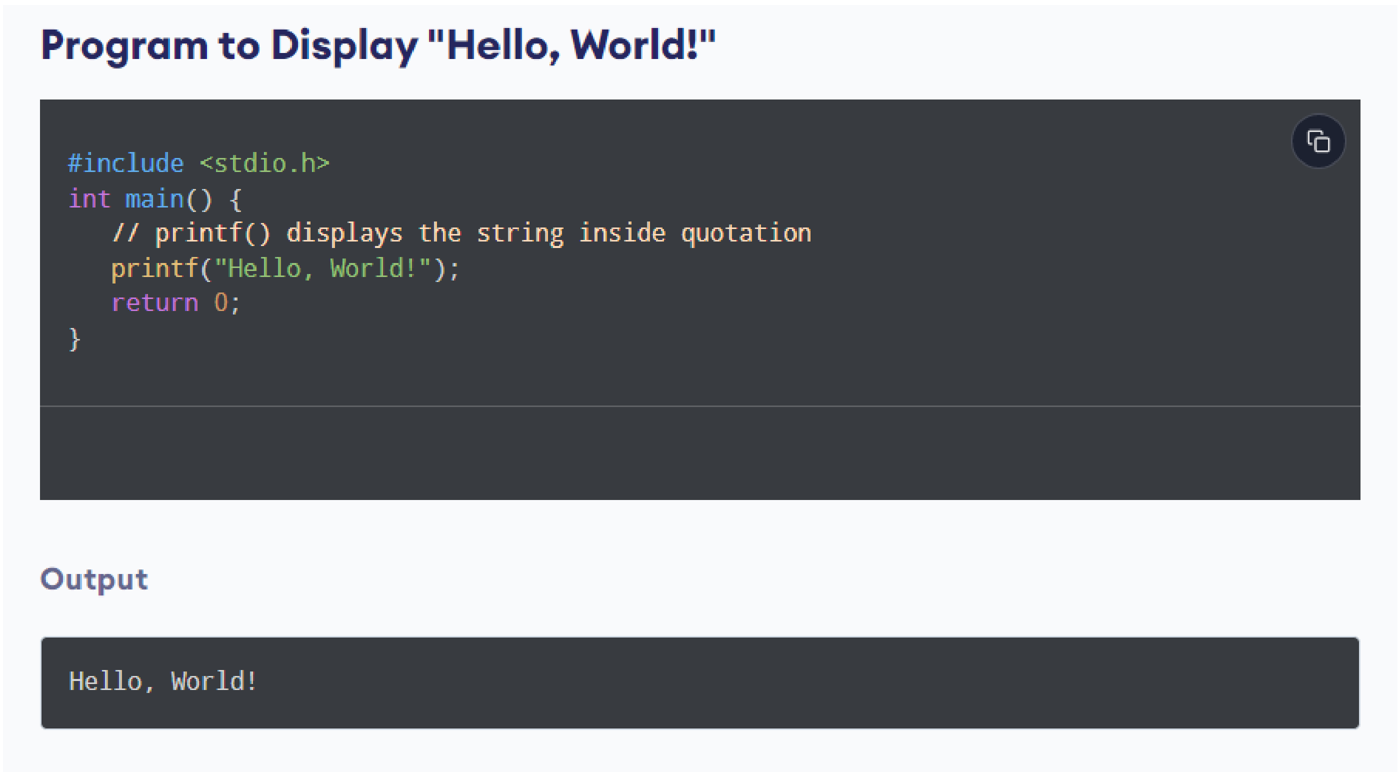
C "Hello, World!" Program

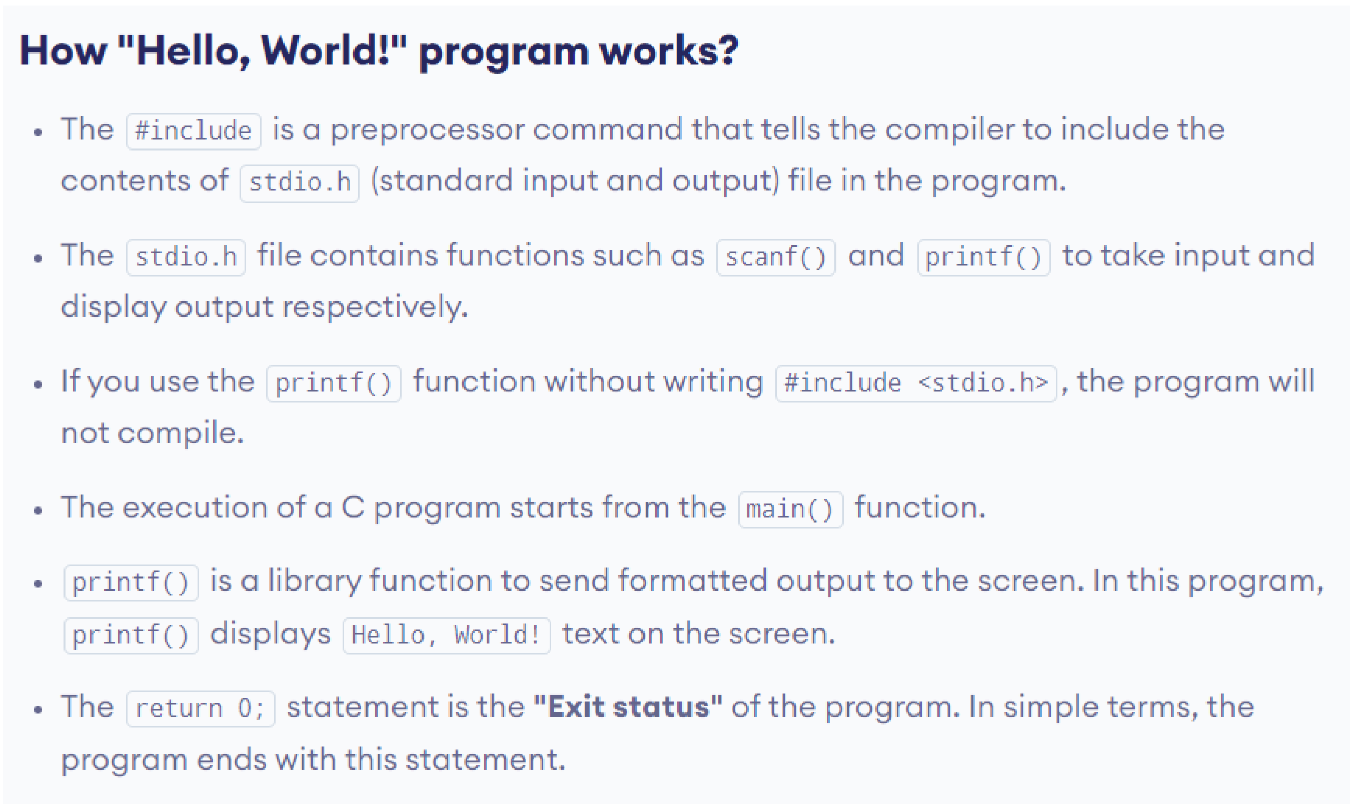


Program to Display "Hello, World!"

```c
#include <stdio.h>
int main() {
   // printf() displays the string inside quotation
   printf("Hello, World!");
   return 0;
}
```

Output

```
Hello, World!
```



How "Hello, World!" program works?

- The `#include` is a preprocessor command that tells the compiler to include the contents of `stdio.h` (standard input and output) file in the program.
- The `stdio.h` file contains functions such as `scanf()` and `printf()` to take input and display output respectively.
- If you use the `printf()` function without writing `#include <stdio.h>`, the program will not compile.
- The execution of a C program starts from the `main()` function.
- `printf()` is a library function to send formatted output to the screen. In this program, `printf()` displays `Hello, World!` text on the screen.
- The `return 0;` statement is the **"Exit status"** of the program. In simple terms, the program ends with this statement.

LAB01-TASK01: Grade Calculator

Instructions:

- Prompt the user to enter their score between 0 and 100.
- Based on the score, calculate, and display the corresponding grade using the following grading scale:

  90-100: A+

  80-89: A

  70-79: B+

  60-69: B

  50-59: C+

  40-49: C

  30-39: D

  Below 30: F
- Display both the score and the grade on the console output.

Requirements:

- Use the "if...else" statement to determine the grade based on the score.
- Implement appropriate error handling to validate the user's input score (ensure it falls within the valid range).
- Ensure the code is properly commented and follows good programming practices.
- Test the code with different input values, including boundary cases, to ensure its correctness.

Sample Output:

```
Enter your score: 87.5
Score: 87.5
Grade: A
```

Rubric:

	Criteria	Points
FUNCTIONALITY	Code compiles without errors and warnings	10
	Code prompts the user for input and validates the score within the valid range (0-100)	10
	Code calculates and displays the corresponding grade based on the score using if…else	20
	Code displays both the score and the grade accurately	5
	Code handles errors and provides appropriate error messages for invalid input	10
READABILITY	Code follows appropriate naming conventions (variables)	5
	Code includes proper indentation and formatting	5
	Code includes comments to explain the logic and enhance understanding	5
	Code is well-structured and organized with meaningful code blocks	5
	Code does not contain unnecessary or redundant statements	5
TESTING	Code produces correct outputs for different test cases, including boundary cases	10
	Code handles various scenarios accurately, such as equal to or near the boundaries	10

LAB01-TASK02: Temperature Classification

You have been assigned an advanced task of creating a program that classifies a given temperature into one of five categories: freezing, cold, moderate, hot, or extreme. Your program should prompt the user to enter a temperature, and then display the corresponding category based on the following rules:

- Freezing: Temperatures below 0 degrees Celsius
- Cold: Temperatures between 0 and 10 degrees Celsius
- Moderate: Temperatures between 11 and 25 degrees Celsius
- Hot: Temperatures between 26 and 35 degrees Celsius
- Extreme: Temperatures above 35 degrees Celsius

Instructions:

- Prompt the user to enter a temperature.
- Implement a switch-case statement to determine the category of the entered temperature based on the rules mentioned above.
- Display the appropriate message indicating the category of the temperature on the console output.

Requirements:

- Use the "case" statement to determine the temperature classification.
- Implement appropriate error handling to validate the user's input (ensure it falls within the valid range).
- Implement appropriate error handling to handle cases where an invalid input (non-numeric input) is entered.
- Ensure the code is properly commented and follows good programming practices.
- Test the code with different input values, including boundary cases, to ensure its correctness.

Sample Output:

```
Enter the temperature in degrees Celsius: 15.5
The entered temperature is moderate.
```

Rubric:

	Criteria	Points
FUNCTIONALITY	Code compiles without errors and warnings	10
	Code correctly prompts the user to enter a temperature	5
	Code accurately determines the category of the entered temperature using a switch-case statement	20
	Code displays the appropriate message indicating the category of the temperature on the console output	5
	Code handles errors and provides appropriate error messages for invalid input	15
READABILITY	Code follows appropriate naming conventions (variables)	5
	Code includes proper indentation and formatting	5
	Code includes comments to explain the logic and enhance understanding	5
	Code is well-structured and organized with meaningful code blocks	5
	Code does not contain unnecessary or redundant statements	5
TESTING	Code produces correct outputs for different test cases, including boundary cases	10
	Code handles various scenarios accurately, such as equal to or near the boundaries	10

LAB01-TASK03: Factorial Calculator

Instructions:

- Prompt the user to enter a positive integer value.
- Calculate and display the factorial of the entered number using the "for" loop.
- Ensure that the entered number is a positive integer greater than or equal to 0.
- Display both the entered number and its factorial on the console output.

Requirements:

- Use "for" loop to calculate the factorial.
- Implement appropriate error handling to validate the user's input (ensure it is a positive integer).
- Ensure the code is properly commented and follows good programming practices.
- Test the code with different input values, including boundary cases, to ensure its correctness.

Sample Output:

```
Enter a positive integer: 5
Number: 5
Factorial: 120
```

Rubric:

	Criteria	Points
FUNCTIONALITY	Code compiles without errors and warnings	10
	Code prompts the user for input and validates that the entered value is a positive integer	10
	Code correctly calculates the factorial of the entered number using either the "for" loop	20
	Code displays both the entered number and its factorial accurately	5
	Code handles errors and provides appropriate error messages for invalid input	10
READABILITY	Code follows appropriate naming conventions (variables)	5
	Code includes proper indentation and formatting	5
	Code includes comments to explain the logic and enhance understanding	5
	Code is well-structured and organized with meaningful code blocks	5
	Code does not contain unnecessary or redundant statements	5
TESTING	Code produces correct outputs for different test cases, including boundary cases	10
	Code handles various scenarios accurately, such as equal to or near the boundaries	10

LAB01-TASK04: Guessing Game

Scenario:

You have been assigned the task of creating a number guessing game program. Your program will generate a random number between 1 and 100, and the user will have to guess the number. The program should provide hints to the user, indicating whether the guessed number is too high or too low. The game will continue until the user correctly guesses the number.

Instructions:

- Generate a random number between 1 and 100 as the target number.
- Prompt the user to enter a guess.
- Implement a while loop that continues until the user correctly guesses the number.
- Within the loop, compare the user's guess with the target number.
- Provide appropriate feedback to the user indicating whether the guess is too high or too low.
- Prompt the user for another guess until they guess the correct number.
- Display a congratulatory message when the user guesses the number correctly.

Requirements:

- Write a program that uses “while” loop.
- Implement appropriate error handling to handle cases where an invalid input (non-numeric input) is entered.
- Ensure the code is properly commented and follows good programming practices.
- Test the code with different inputs to ensure its correctness.

Sample Output:

```
Welcome to the Number Guessing Game!
Guess a number between 1 and 100: 50
Too low! Guess a higher number: 75
Too high! Guess a lower number: 60
Too low! Guess a higher number: 65
Congratulations! You guessed the number 65 correctly.
```

Rubric:

	Criteria	Points
FUNCTIONALITY	Code compiles without errors and warnings	5
	Code generates a random number between 1 and 100 as the target number	10
	Code prompts the user to enter a guess and handles user input correctly	5
	Code implements a while loop that continues until the user correctly guesses the number	15
	Code provides appropriate feedback to the user indicating whether the guess is too high or too low	10
	Code displays a congratulatory message when the user guesses the number correctly	5
READABILITY	Code follows appropriate naming conventions (variables)	5
	Code includes proper indentation and formatting	5
	Code includes comments to explain the logic and enhance understanding	5
	Code is well-structured and organized with meaningful code blocks	5
	Code does not contain unnecessary or redundant statements	5
TESTING	Code produces correct outputs for different test cases, covering different target numbers and user guesses	20
	Code handles various scenarios accurately, such as equal to or near the boundaries	10

LAB01-TASK05: Scenario Based Question

Scenario:

You are tasked with creating a HR program that generates employee ID AND calculating an employee's age based on the birthdate entered.

For the birthdate:

Your program should prompt the user to enter their birthdate in the format of day, month, and year. It should then calculate their age accurately and display it on the console output.

For the employee ID:

The IDs should follow a specific format where the prefix is always "R" followed by a random 4-digit number.

Instructions:

- Prompt the user to enter their birthdate.
- Implement a user-defined function called calculateAge that takes the birthdate as input and returns the calculated age as an integer.
- The function should use the current date to calculate the age accurately, considering leap years and varying month lengths.
- Display the calculated age on the console output.
- Implement a user-defined function called generateEmployeeID that takes no arguments and returns a string representing a randomly generated employee ID.
- The function should generate a random 4-digit number and concatenate it with the prefix "R" to form the employee ID.
- Display the generated employee ID on the console output.

Requirements:

- Write a user-defined function calculateAge that accepts the birthdate as input and returns the calculated age as an integer.
- Implement appropriate error handling to handle cases where an invalid birthdate or future date is entered.
- Ensure the code is properly commented and follows good programming practices.
- Test the code with different birthdates to ensure its correctness.
- Write a user-defined function generateEmployeeID that generates a random employee ID following the specified format.
- Ensure that the generated employee ID is unique for each run of the program.
- Implement appropriate error handling to handle any potential issues during the ID generation process.
- Ensure the code is properly commented and follows good programming practices.
- Test the code to ensure it generates random and unique employee IDs.

Sample Output:

```
Enter name: Mohd Amirul Najmuddin
Enter department: Accounts
Enter birthdate (day month year): 15 03 1990

Employee Details
------------------------
Employee ID: R8567
Name: Mohd Amirul Najmuddin
Department: Accounts
Age: 33
```

Rubric:

	Criteria	Points
FUNCTIONALITY	Code compiles without errors and warnings	5
	Code prompts the user to enter their birthdate and handles invalid input	10
	Code correctly implements the user-defined function calculateAge to calculate the age based on the birthdate	15
	Code displays the calculated age accurately on the console output	10
	Code correctly implements the user-defined function generateEmployeeID to generate a random employee ID with the specified format	15
	Code handles errors and provides appropriate error messages for invalid input	5
READABILITY	Code follows appropriate naming conventions (variables)	4
	Code includes proper indentation and formatting	4
	Code includes comments to explain the logic and enhance understanding	4
	Code is well-structured and organized with meaningful code blocks	4
	Code does not contain unnecessary or redundant statements	4
TESTING	Code produces correct outputs for different test cases, including boundary cases	10
	Code handles various scenarios accurately, such as equal to or near the boundaries	10

Lab Task Submission:

1. LAB01-TASK01.c
2. LAB01-TASK02.c
3. LAB01-TASK03.c
4. LAB01-TASK04.c
5. LAB01-TASK05.c

Zip all your files and save it as LAB01_<GroupName>_<Day>.zip

Example: LAB01_Group1_WED.zip

Group Name
Group1 – Group15

Day
TUES
WED
THURS