Certainly! Let's break **Question 1: 'C' Basics** into detailed sub-topics for better understanding:

---

1. History of C

- Developed by **Dennis Ritchie** in **1972** at Bell Labs.
- C evolved from two previous languages: **B** and **BCPL** (Basic Combined Programming Language).
- Initially designed for system programming (e.g., operating systems like Unix).
- Known as the **mother of all programming languages** because many modern languages like C++, Java, Python, etc., are derived from it.

---

2. Structure of a C Program

A C program is composed of:

- **Preprocessor directives**: Start with `#` (e.g., `#include <stdio.h>` for standard input/output library).
- **Main function**: Entry point of a program, `int main()`.
- **Statements/Functions**: Code instructions inside `{}`.
- Example:

```
#include <stdio.h> // Preprocessor directive
int main() {       // Main function
    printf("Hello, World!"); // Output statement
    return 0;       // Exit code
}
```

---

3. Compilation Process

- **Steps**:
    1. **Preprocessing**: Handles `#include` and macros.
    2. **Compilation**: Converts C code to Assembly code.
    3. **Assembly**: Converts Assembly code to Machine code.
    4. **Linking**: Links libraries and creates the final executable.
- Tools: GCC (GNU Compiler Collection) is commonly used.

---

4. Variables and Constants

- **Variable**: A name that stores data. Syntax: `data_type variable_name;`
  Example:

```
int age = 20;  // Declaring and initializing an integer variable
```

- **Constant**: Fixed value that cannot be changed. Declared using `const`.
  Example:

```
const float PI = 3.14;  // Declares a constant PI
```

---

5. Keywords and Identifiers

- **Keywords**: Reserved words in C (e.g., `int`, `return`, `if`). Total: **32** in standard C.
- **Identifiers**: Names used for variables, functions, or arrays.
  - Must start with a letter or `_`.
  - Cannot use spaces or special characters.

---

6. Input/Output Functions

- **Input**: Use `scanf()` to take input from the user.
  Example:

```
int num;
printf("Enter a number: ");
scanf("%d", &num);  // Reads an integer
```

- **Output**: Use `printf()` to display data.
  Example:

```
printf("The number is: %d", num);  // Outputs the number
```

Key Points to Remember

- C is case-sensitive (`Variable` and `variable` are different).
- Every statement ends with a semicolon (`;`).
- Use comments for clarity:
  - **Single-line**: `// Comment`
  - **Multi-line**: `/* Comment */`

Do you need further clarification on any of these topics or example programs?