Module: R1: C Programming
Section: C Inline Functions, Structures **Task:** 2.2

Task 3.1
Inline Assembly

Intro:

This C program performs integer addition using inline assembly. The program prompts the user to enter two integers, reads the input values, adds them together using inline assembly, and then prints the result.

Explanation:

The program begins by declaring two integer variables, `x` and `y`, to store the user input. It prompts the user to enter the first integer and reads the input using **`scanf()`**. Similarly, it prompts the user to enter the second integer and reads the input into the variable `y`. The core of the program lies in the inline assembly code block:

```
__asm__ volatile("add %1, %0": "+r" (x) : "r" (y) : "cc");
```

In this assembly block:

1. **`add`** is the assembly instruction to perform addition.
2. **`%1`** represents the second operand (`y`), and **`%0`** represents the first operand (`x`).
3. The "**`+r`**" constraint indicates that `x` is both an input and an output operand, while "**`r`**" indicates that `y` is an input operand.
4. "`cc`" informs the compiler that the operation may affect the condition codes.
5. After performing the addition operation, the program prints the result using **`printf()`**.

Code Snippet:

```
/*
* Author: Noman Rafiq
* Dated: June 25, 2024
* Description: The programs takes two inputs (x & y) from the user and outputs the sum of them using inline assembly.
*/
```

```c
#include<stdio.h>

int main(){
	unsigned int x,y;

	//Takes x
	printf("Enter x: \n");
	scanf("%u", &x);

	//Takes y
	printf("Enter y: \n");
	scanf("%u", &y);

	//Inline Assembly
	__asm__ volatile ("add %1, %0" : "+r" (x) : "r" (y) : "cc");

	printf("Sum is = %u\n", x);
}
```