

1. Implement a C Program for Reversing a 32 bit signed integers

Code

```
#include <stdio.h>

#include <limits.h> // For INT_MAX and INT_MIN

int reverse(int x) {
    long reversed = 0;

    while (x != 0) {
        int digit = x % 10;
        x /= 10;

        reversed = reversed * 10 + digit;

        // Check for overflow
        if (reversed > INT_MAX || reversed < INT_MIN)
            return 0;
    }

    return (int)reversed;
}

int main() {
    int num = 1234; // Example number
    int result = reverse(num);

    printf("Original number: %d\n", num);
    printf("Reversed number: %d\n", result);
```

```
    return 0;  
}  
}
```

Output

The screenshot shows a code editor interface with a tab labeled "main.c". The code implements a function to reverse the digits of an integer. It includes necessary header includes, a check for overflow, and a main function demonstrating the usage.

```
1 #include <stdio.h>  
2 #include <limits.h> // For INT_MAX and INT_MIN  
3  
4 int reverse(int x) {  
5     long reversed = 0;  
6  
7     while (x != 0) {  
8         int digit = x % 10;  
9         x /= 10;  
10  
11         reversed = reversed * 10 + digit;  
12  
13         // Check for overflow  
14         if (reversed > INT_MAX || reversed < INT_MIN)  
15             return 0;  
16     }  
17  
18     return (int)reversed;  
19 }  
20  
21 int main() {  
22     int num = 1234; // Example number  
23     int result = reverse(num);  
24  
25     printf("Original number: %d\n", num);  
26     printf("Reversed number: %d\n", result);  
27  
28     return 0;  
29 }
```

The "Run" button is highlighted. To the right, the "Output" pane displays the results of the program's execution:

```
Original number: 1234  
Reversed number: 4321  
==== Code Execution Successful ===
```